Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di Laurea Magistrale in Informatica

Zero-shot Cross-language Relation Extraction as Multilingual Machine Comprehension

Relatore: Dott. Elisabetta Fersini

Correlatore: Prof. Anders Søgaard

Relazione della prova finale di:
Cezar Angelo Sas
Matricola 781563

Anno Accademico 2018-2019

Contents

Chapter 1

Introduction

With the advent of the internet, a large amount of digital text is produced each day as news articles, research publications, blogs, and social media. It is essential to advance techniques for automatically extracting information from these documents, as much valuable knowledge is hidden within them. This extracted information can be used to enhance access and management of knowledge hidden in large text corpora. The computer science field that concerns the processing of natural language is called Natural Language Processing (NLP).

However, most of this information is produced only in English; indeed, it is a widely lamented fact that linguistic and encyclopedic resources are heavily biased towards English. Even multilingual knowledge bases (KBs) such as Wikidata (Vrandečić and Krötzsch, 2014) are predominantly English-based (Kaffee and Simperl, 2018). This suggests that coverage is higher for English and that facts of interest to English-speaking communities are more likely included in a KB.

This also means that current natural language processing research and development has mainly focused on English. Recognizing this, the community has in recent years put emphasis on multilingual and cross-lingual resources and methods which allow for the joint exploitation of resources between languages and for the transfer of models from well-resourced languages to lower-resourced ones.

In this work, we discuss such problems in the context of Relation Extraction (RE), which is an Information Extraction (IE) subtask. IE is a field that concerns NLP, computational linguistic, and data mining. The goal of IE is to extract structured information from a collection of unstructured documents, this is achieved via different subtask like named entity recognition, coreference resolution, and relation extraction. Relation extraction consists of identifying mentions of the relations of interest in each sentence of the given documents, the extracted relation are then used to populate knowledge bases. Traditional relation extraction system are required to only extract seen relations, which make easier to train these models as the data required can be obtained via crowd sourcing (Liu et al., 2016) or distant

supervision (Hoffmann et al., 2011). These models are meant to generalize to new entities, but *not* new *relations*. In this work, we consider the case in which models are trained on a subset of pre-specified relations and applied to both seen and unseen entities, and unseen relations. The latter scenario is known as *zero-shot* RE (Rocktäschel et al., 2015).

As a possible solution to this problem, Levy et al. (2017) propose an alternative approach to the relation extraction task. By reframing the problem as question answering, they can use reading comprehension techniques that allow zero-shot relation extraction.

Our interest in multilingual relation extraction concerns three different questions:

a) how well relation extraction models can be transferred across different languages?
b) can the variance in the number of relations examples between languages help build more robust relation extraction models?
c) can a jointly trained model that performs RE on multiple languages perform equally or better than the monolingual trained version?

Unfortunately, as aforementioned, there is a lack of multilingual resources, and this is also true for relation extraction. For the best of our knowledge, there is no large corpora for multilingual relation extraction available. The only resource contains around 90K examples and it is available only in three languages. To solve this and answer our questions, based on (Levy et al., 2017; Hewlett et al., 2016) we propose **X-WikiRE**, a new multilingual resource for relation extraction as question answering and using state-of-the-art models we investigate how such resource can be used in different transfer learning scenarios. By performing various experiments we show how our multilingual resource helps both in case of low resource languages by performing sequential transfer learning, and also that **X-WikiRE** improves the performances on single languages when jointly training a single model on multiple languages.

This work is organized through the following chapters: in Chapter 2 we present some natural language processing and machine learning concepts and methodologies background. In Chapter 3 we present the state of the art in relation extraction and machine comprehension. Next, Chapter 4 presents **X-WikiRE**, our new multilingual relation extraction dataset; while in Chapter 5 we describe the used machine comprehension model and our experimental setup. Finally, in Chapter 6 we present the results obtained using our dataset. To conclude, Chapter 7 contains some final thoughts and future developments.

Chapter 2

Background

This chapter provides an overview of the different tasks and concepts needed to perform zero-shot relation extraction in a multilingual setup. Some concepts, like word representation (Section 2.3) are at the core of NLP, while others are more general machine learning techniques (Section 2.1, 2.4 and 2.5), others instead are both general and have a very high usage in NLP due to their nature of modeling the intrinsic sequential nature of natural languages (Section 2.2).

2.1 Neural Networks

A Neural Network (NN) is a computational model inspired by networks of biological neurons, wherein the neurons compute output values from inputs. The simplest form of neural network is the feed forward neural network, in which the information flows only in one direction, and there is no recurrent connection between the nodes. Neural networks try to approximate some function f^* by defining a mapping $y = f(x; W)$ for an input x by learning the value of the parameters W that yield the best function approximation. A neural network is called network because it contains multiple layers of artificial neurons. The artificial neuron is the building block of neural networks.

A neuron is defined by an input x, a set of weights W, a bias b, and an activation function g. The activation function should be differentiable, since to actually learn, we need to compute derivatives. With $x, W \in \mathbb{R}^{n \times 1}$, $b \in \mathbb{R}$. The output y is defined as:

$$y = g(W^\intercal x + b) \tag{2.1}$$

Feed forward neural networks are composed by various layers, each layer contains some neurons. The formal definition is very similar to the one of the neuron. To simplify the notation, we include the bias term in the weights matrix, and have

an extra element $x_0 = 1$ in the input x. Given a feed forward neural network with L layers, the forward propagation is:

$$\begin{aligned} z^i &= W^{i-1} a^{i-1} \\ a^i &= g(z^i) \\ y &= a^L g(z^L) \end{aligned} \tag{2.2}$$

Where g is an activation function, a^i is the activation of layer i, and $a^0 = x$. The weight matrix W_i controls the projection from layer i to layer $i+1$. Given a network with s_i units in layer i, and s_{i+1} in layer $i+1$ the weight matrix $W_i \in \mathbb{R}^{(s_{i+1}) \times (s_i+1)}$.

As in other machine learning algorithms, we have to learn the parameters of our neural network. We have a cost function, or objective function, $J(\theta)$ that we try to minimize. Given a neural network h_θ, we compute our cost function as a sum of the error function $\mathcal{L}$ for all the examples. The error function computes the difference between the predicted value $\hat{y} = h_\theta(x)$ and the actual value y. We also have a regularizer term Ω that is weighted by the hyperparameter λ:

$$\min_\theta J(\theta) = -\frac{1}{N} \sum_1^N \mathcal{L}(y_i, \hat{y}_i) + \lambda \Omega_\theta \tag{2.3}$$

Now that we have defined what is the goal of our neural network, we can start performing the training. The weights θ in the network are the only parameters that can be modified to make the cost function J as low as possible; thus we can minimize J by using an iterative process of gradient descent, for which we need to calculate the gradient for the cost function with respect to the network parameters. As each network consists of various layers, computing the gradient of the loss function w.r.t the parameters $\frac{\partial J}{\partial \theta}$ is non-trivial. To estimate the gradient, an algorithm called backpropagation (Rumelhart et al., 1988), from backward propagation of errors, is used. As the name implies, backpropagation starts computing the gradients from the output of the networks and moves backward to the input through all layers. First, we need to compute the derivative of $J(\theta)$ w.r.t the output, this will be our ∂^L, then go backward:

$$\begin{aligned} \partial^L &= \frac{\partial}{\partial y} \mathcal{L}(y, \hat{y}) \\ \partial^i &= \theta^{i\intercal} \partial^{i+1} \odot \left(\frac{\partial}{\partial z^i} g(z^i) \right) \end{aligned} \tag{2.4}$$

For each layer, the computed gradients are then used to update the corresponding parameters using gradient descent.

2.2 Sequence Modeling

Due to the inherent nature of natural language almost all NLP tasks deal with sequences. For example, in speech recognition, we have the input audio which is a sequence of phonemes that will be recognised and used to produce a transcript. Likewise, in machine translation we have as input a sequence of words in a source language and the machine has to generate a new sequence of words with the same meaning of the input sequence but in a target language.

Traditional neural networks are not designed to deal with the temporal information of sequences in an efficient way. A way to handle this problem is to introduce the notion of a dynamical system. In the dynamical system, there is a state which carries information about past inputs. Typically, we make what is called a Markov assumption; that means that all the information regarding the past and current inputs required to estimate a future state is succinctly contained in the current state. A Recurrent Neural Network (RNN) is a particular type of dynamical system.

2.2.1 Recurrent Neural Network

The most elementary neural network for sequential input is the recurrent neural network (Elman, 1990). Elman proposes a recurrent network based on the idea introduced by Jordan (1986) which proposes a network with recurrent connections from the output to the input; these are one-to-one connections that are used to associate a static pattern with a serially ordered output pattern. Elman modifies the network by moving the recurrent connection. Instead of having the output connected with the input, Elman (1990) adds additional units at the input level, these units are called "*context units*" and are connected to the hidden layer. Context units are also "hidden" in the sense that they interact exclusively with other nodes internal to the network, and not the outside world.

In Figure 2.1 we can have a look at how a RNN works. The network A takes in input x_t and produces an output h_t, a loop enables the information to flow from the previous step to the current, allowing the prediction to be affected by the last state of the network. A more effective way to think about RNNs is by unrolling them (right side of Figure 2.1). We can formally define A as:

$$\begin{aligned} h_t &= f_W(h_{t-1}, x_t) \\ y_t &= g_\theta(h_t) \end{aligned} \tag{2.5}$$

Where, h_t is the new state, f_W and g_θ are functions with parameters W and θ respectively, h_{t-1} is the previous state, x_t is the current input, and y_t is the output

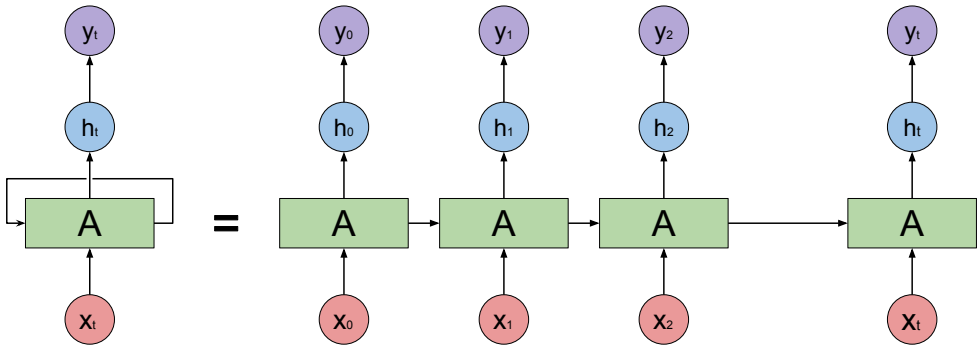


Figure 2.1: RNN unrolling. On the left a compact representation of RNNs, on the right the same RNN can be unrolled

at the *t-th* step. The vanilla RNN is defined using a neural network with tanh as activation function, so we have:

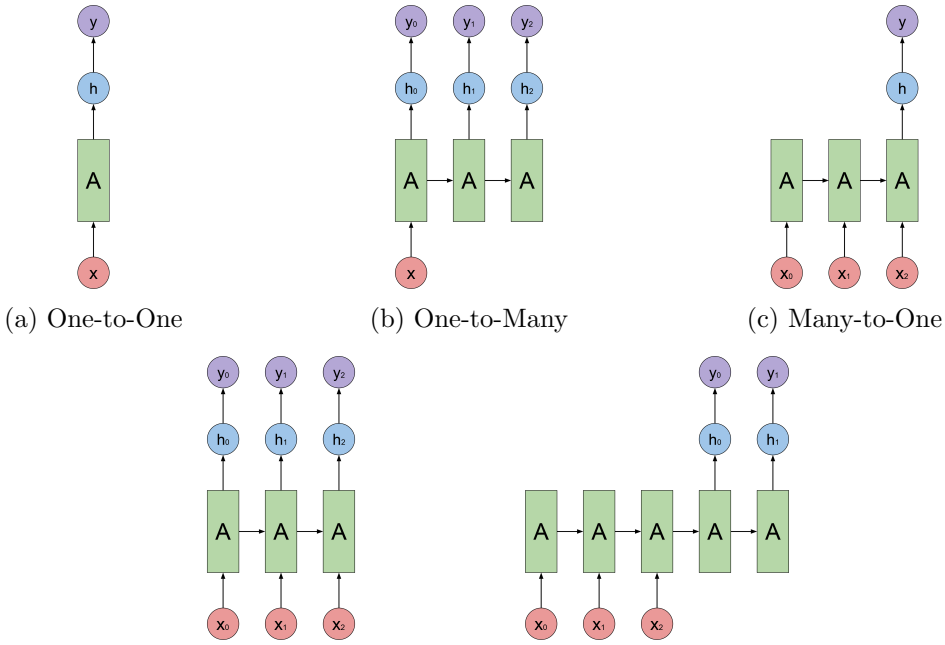
$$\begin{aligned} h_t &= \tanh\left(W_h \cdot h_{t-1} + W_x \cdot x_t\right) \\ &= \tanh\left([W_h, W_x] \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}\right) \\ &= \tanh\left(W \cdot \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix}\right) \end{aligned} \tag{2.6}$$

Where W_h, W_x are weights for the hidden and input respectively.

RNN are able to manage variable length of input and output sequences, we can have different types of RNN according to these sizes. Given a sequence x of length l_x, and a desired output sequence y with length l_y, we can define the following types (visually represented in Figure 2.2):

a) One-to-One: $l_x = l_y = 1$, this is a traditional neural network;
b) One-to-Many: $l_x = 1, l_y > 1$, this is the case for sequence generation;
c) Many-to-One: $l_x > 1, l_y = 1$, for sequence classification;
d) Many-to-Many: $l_x = l_y > 1$, for sequence labeling;
e) Many-to-Many: $l_x \neq l_y, l_x > 1, l_y > 1$, also referred to as sequence-to-sequence (Sutskever et al., 2014), or encoder-decoder (Cho et al., 2014b). Given an input sequence, the encoder process it and generates a dense representation which is used to initialize the decoder that will produce the output sequence. This is elaborated in Section 2.2.3

We can also have bidirectional RNNs (Schuster and Paliwal, 1997) (BiRNN). A BiRNN is composed of two RNNs, one looks at the sequence starting from the



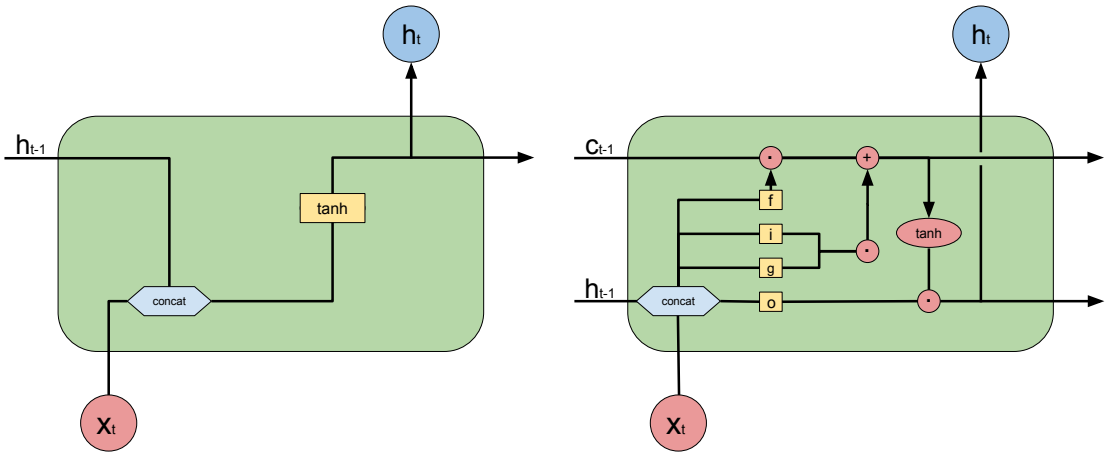
(a) One-to-One (b) One-to-Many (c) Many-to-One

(d) Many-to-Many (e) Many-to-Many

Figure 2.2: Different types of RNN

end while the other from the beginning as in the vanilla RNN. The output at each step of a bidirectional RNN is the concatenation of the hidden states for that time step. In a BiRNN we have a left-to-right hidden state (or forward state) $\overrightarrow{h_t} = RNN(x_t, \overrightarrow{h}_{t-1})$, a right-to-left one (or backward state) $\overleftarrow{h_t} = RNN(x_t, \overleftarrow{h}_{t-1})$, and the output hidden state h_t is defined as the concatenation of the directional hidden layers: $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. The advantage of having bidirectional information is quite clear, the model is able to represent also future information, so the current decision is affected by both past and future.

Recurrent networks have various advantages over traditional NN. These include the ability to process sequences of any length, the size of the model does not increase with the size of the input, and most important, the computation takes into consideration historical information that should allow RNNs to learn long-term dependencies. Unfortunately, they also have some drawbacks. They are computationally more intensive, and even if they can work with sequences of any length and pass information through time, this information transfer starts to become less effective for longer sequences, which means they struggle to model longer-term dependencies. That is caused by the vanishing/explosion of the gradient, a significant issue in vanilla RNN.



(a) Vanilla RNN cell (b) LSTM cell

Figure 2.3: Different types of cells. Red circles are for point-wise operations, in yellow we have neural networks.

The vanishing gradient problem was first studied by (Hochreiter, 1991; Bengio et al., 1994). The problem occurs when we try to train a neural network model using gradient-based optimisation techniques, the backpropagation of the loss function with respect to the weights tends to vanish. This is a consequence of what happens during forward propagation; the hidden state is multiplied several times with the weight matrix, once per time step. So, during backpropagation, this leads to the gradients being multiplied by the same values many times, causing the gradient values to either explode, i.e. grow extremely large or vanish, i.e. become very small, making the model unable to learn. In RNNs this translates to that is very tough to allow the error to flow from the further in time inputs to the ones at the beginning, thus creating difficulties to train the early stages of the RNN and reducing the ability to learn long-term dependencies due to insufficient weight changes. A solution to this problem is the use of a more complex logic for updating the internal state of the cell, for example as in the LSTMs.

2.2.2 Long Short Term Memory

Long Short Term Memory networks (LSTM) are an improved version of RNNs that introduce mechanisms to decide what should be "remembered" and "forgotten". Proposed by Hochreiter and Schmidhuber (1997), LSTMs are a solution to the long-term dependencies issue in vanilla RNN and the vanishing of the gradient. LSTMs mitigate the problem by having a more complex cell that contains two hidden states. The first one is h_t and is similar to the one in RNNs, the other

is C_t, the cell state, which is the core of a LSTM cell. The cell state is modified using the gates, which are neural networks acting on the information of the cell state and the hidden state. In Figure 2.3 we can see how the information flows and interact with the different gates, these are:

- Forget gate layer (f): acts on what and how much to forget from the previous time step. It is computed using a neural network with a sigmoid σ activation function. The output of the network is between 0 and 1, since the output will be multiplied element-wise 0 means forget all, and 1 remember all. $f_t = \sigma(W_f * [h_{t-1}, x_t])$;

- Input gate layer (i): this gate decides which information to write to the cell state. It is a sigmoid layer: $i_t = \sigma(W_i * [h_{t-1}, x_t])$, and works in combination with the gate layer;

- Gate gate layer (g): decides how much to write to the cell state. It is a tanh layer, $g_t = \sigma(W_g * [h_{t-1}, x_t])$. The output is the multiplied by the input gate's output to create a new candidate state cell that will be summed with the value of the previous cell state after the forget step. The new cell state is $c_t = f_t \odot c_{t-1} + i_t \odot g_t$;

- Output gate layer (o): this layer is responsible of how much of the cell state to reveal through the hidden state. This is done using a sigmoid layer which decides the parts of the cell state that will be shown in the output, formally $o_t = \sigma(W_o * [h_{t-1}, x_t])$. Then, we put the cell state through tanh (to push the values to be between -1 and 1) and multiply it by the output's layer output, so that we only expose the parts we decided to. Formally, the hidden state will be $h_t = o_t \odot \tanh(c_t)$.

This is the general LSTM; however there are many variants of LSTMs, which perform slightly better or worse depending on the task. A more detailed work on how each variant perform is "LSTM: A Search Space Odyssey" by Greff et al. (2017).

2.2.3 Attention Mechanism

A very popular RNN architecture is the sequence-to-sequence, or encoder-decoder (Cho et al., 2014b; Sutskever et al., 2014). This framework is very popular in many NLP tasks, including neural machine translation, text summarization, question answering, and many more.

A sequence-to-sequence model is composed by an encoder, which takes the input sequence and compresses it into a context vector which will be used to

start the decoder which predicts an output sequence. A significant and clear disadvantage of a fixed-size context vector design is as the sequence gets longer and longer, more memory is required to store past information but the encoder is unable to represent all of it due to the fixed dimension of the vector. As a result, the network performs poorly on longer sequences (Cho et al., 2014a). A solution to this has been proposed by Bahdanau et al. (2015), by introducing an attention mechanism for the decoder, instead of relying only on the previous hidden state, the decoder with attention can use information from all the input sequence by looking at the encoder hidden state at each time step and combine this information to produce the output.

Encoder: Given a sequence of vectors $\mathbf{x} = (x_1, \ldots, x_n)$, an RNN takes this sequence and produces a context vector c:

$$\begin{aligned} h_t &= f(x_t, h_{t-1}) \\ c &= q(\{h_1, \ldots, h_n\}) \end{aligned} \tag{2.7}$$

where h_t is the hidden state at time t, and f and q are nonlinear functions. Some example of f and g include: LTSMs for f, and return the last hidden state for g. This part is the same with or without attention mechanism.

Decoder: Given a context vector, the decoder computes the probability of a sequence $\mathbf{y} = (y_1, \ldots, y_T)$ by decomposing the joint probability into the ordered conditionals to the context vector, and the previous outputs.

$$P(\mathbf{y}) = \prod_{t=1}^{T} P\left(y_t | \{y_1, \cdots, y_{t-1}\}, c\right) \tag{2.8}$$

Which can be implemented using RNNs, where g is a nonlinear function, and s_t is the decoder hidden state:

$$P(\mathbf{y}) = \prod_{t=1}^{T} g\left(y_{t-1}, s_t, c\right) \tag{2.9}$$

Decoder with Attention: Illustrated in Figure 2.4, the Bahdanau et al. (2015) decoder performs the same task as the before, but instead of being conditioned to the context vector and the previous output, the probability of each output is conditioned to the previous outputs, and the entire input sequence.

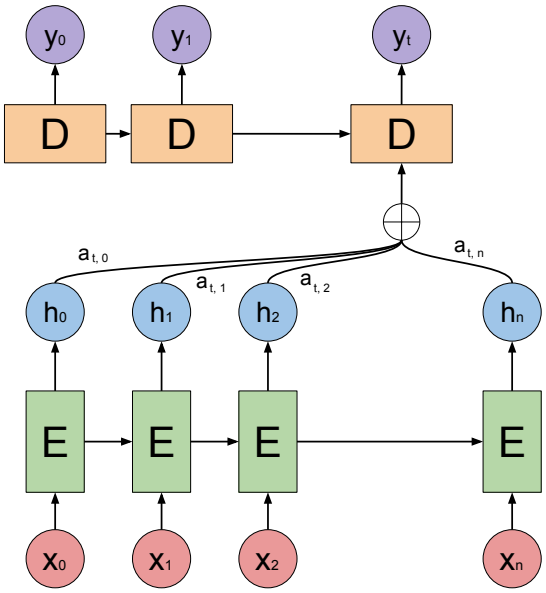


Figure 2.4: Attention mechanism. The encoder E hidden states are weighted via $a_{i,j}$, summed, and used as input to the decoder D to produce the output at step t.

$$P(\mathbf{y}) = \prod_{t=1}^{T} P\left(y_t | \left\{y_1, \cdots, y_{t-1}\right\}, \mathbf{x}\right) = \prod_{t=1}^{T} g\left(y_{t-1}, s_t, c_t\right) \tag{2.10}$$

The new context vector is computed as a weighted sum of the encoder hidden state $(h_1, \ldots, h_n)$:

$$\begin{aligned} c_i &= \sum_{j=1}^{n} \alpha_{ij} h_j \\ \alpha_{ij} &= \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{n} \exp\left(e_{ik}\right)} \\ e_{ij} &= a\left(s_{i-1}, h_j\right) \end{aligned} \tag{2.11}$$

The weight α_{ij} is a probability that indicates the importance of the j-th encoder state h_j with regard to the previous decoder hidden state s_{i-1} to produce the next hidden state s_i and the output of y_i . The weights are computed using a neural network a called alignment model, since Bahdanau et al. (2015) proposed this architecture in a neural machine translation task, and the idea behind it is to find which words in the source sequence are more relevant to predict the output by effectively learning to align the source sentence with the target one. Having

the decoder look at the entire input sequence implements a so-called attention mechanism, which allows the encoder to decide at which part of the input to attend. By letting the decoder have this mechanism, we reduce the encoder responsibility of having to encode all information in the source sequence into a fixed length vector since the decoder can selectively retrieve the useful information in the sequence accordingly to the current state needs.

Even if attention can be used as an interpretation method for machine translation, Jain and Wallace (2019) used an adversarial setup and found that there are some tasks like text classification, question answering and natural language inference where the gradient based feature importance measure is not equivalent to the attention weights, and it is also possible to train new weights of the attention mechanism while maintaining the same performance and having these new weighs mean nothing to human experts.

The idea of having an attention mechanism became very successful in many other NLP task due to the increase in performance it brings. Its value in NLP inspired a new model, called Transformer (Vaswani et al., 2017), which separates the attention mechanism from the RNN, having a framework entirely based on it.

2.2.4 Transformer

Sequence-to-sequence frameworks are state-of-the-art in most of sequence transduction tasks, and almost all of these models use attention mechanism. However, they need sequential computation due to their nature, making models hard to train. Vaswani et al. (2017) proposes a new framework, the transformer, that still has the same overall encoder-decoder structure, but without recurring networks, allowing their model to be paralleled and making it less needy in terms of resource and computational time while also being able to set new state-of-the-art performance in various tasks.

Vaswani et al. (2017) also generalize the idea of attention by proposing a new formalization. According to them, the attention is a function that maps a query Q and a set of key-value pair $\langle K, V \rangle$ to an output. The output is a weighted sum of the values. The weight is assigned by computing the compatibility function of the query with the corresponding key. In terms of the definition of Bahdanau et al. (2015) we have that the query is the decoder hidden state (s_j), the keys are the encoder hidden state (h_j), and the values are the hidden states of the encoder. Vaswani et al. (2017) uses a so called scaled-dot product attention (Equation 2.12), while the one that used by Bahdanau et al. (2015) is called additive or concat attention, but there are many others.

$$Attention\,(Q, K, V) = softmax\left(\frac{QK^{\intercal}}{\sqrt{d_k}}\right)V \tag{2.12}$$

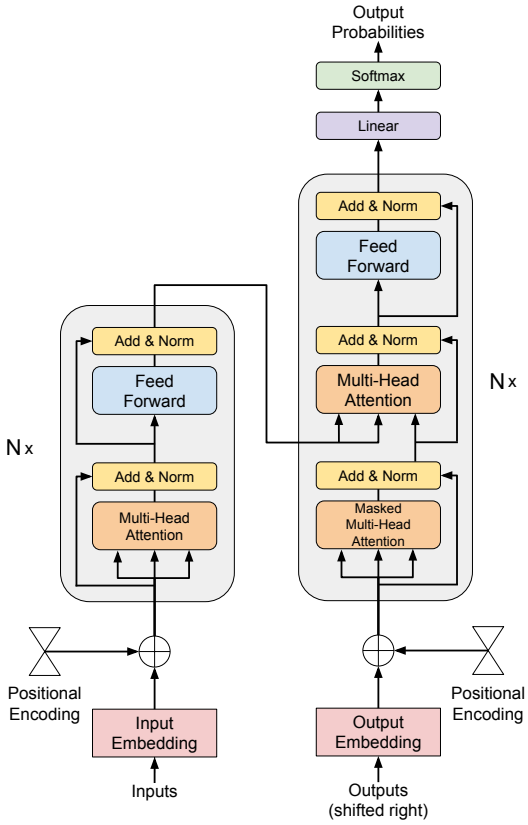


Figure 2.5: Transformer model architecture.

The transformer has a stack of N encoders (left part of model in Figure 2.5), and also N stacks of decoders (right side of Figure 2.5). A stack is made up by different sub-layers.

Encoder: it has two sub-layers, a multi-head self-attention one and a feed forward neural network. Each sub-layer has residual connection (He et al., 2016) and, layer normalization (Ba et al., 2016). The output of each sub-layer is $LayerNorm(x + Sublayer(x))$.

Decoder: it has the same sub-layers as the encoder plus an extra one that performs multi-head attention over the encoder output. The first attention layer is changed to perform a masked version of attention, forcing the prediction at time t to be dependant only on the know outputs.

A big part of the transformer is the multi-head attention (Equation 2.13). The transformer computes h different attentions. The attentions are computed by

projecting the query Q, the keys K, and the values V h different times resulting in h different attention heads exhibiting different behaviours that seems related to the sentence structure making each head learn a different task (e.g. anaphora resolution). In (Voita et al., 2019) they showed how, depending on the task, not all heads are necessary for achieving good performance, and they can be pruned based on the specialization they have. The heads are then concatenated and projected one last time to get the final result. There are two types of attention in the transformer: the regular encoder-decoder attention, and the self-attention. Self-attention, or intra-attention (Cheng et al., 2016), is an attention mechanism relating different positions of a single sequence in order to compute a representation of it, it is used between the sub-layers of both the encoder and the decoder, allowing all the position to attend over all other tokens in the sequence.

$$\begin{aligned} MultiHead(Q, K, V) &= Concat\left(head_1, \ldots, head_h\right) W^O \\ head_i &= Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{2.13}$$

There are state-of-the-art solutions that use the transformer as their construction core, some of these solution are beating multiple NLP challenges without changes to their architecture. A couple examples are BERT (Devlin et al., 2019), and OpenAI GPT (Radford et al., 2018, 2019).

The transformer has also some major drawback, the perk of being parallelizable comes at the cost of loosing the recurrent inductive bias of RNNs, which seems to be essential for generalizing on different sequence modeling tasks of varying complexity. The transformer is also not Turing complete, and lacks conditional processing of the input sequence. Some proposed solution are the Transformer-XL (Dai et al., 2019) and the Universal Transformer (Dehghani et al., 2018).

2.3 Word Representation

Word representation is a fundamental part of most Natural Language Processing (NLP) tasks. The understanding of text is mainly derived by transforming text into usable computational representations, these include graphs, trees, or vectors on which will the focus. In general, it has been found to be beneficial to represent words or documents as vectors, which have an appealing, intuitive interpretation, and they capture hidden information concerning a language, like word analogies or semantic.

A very simple way to transform words into vectors is the one-hot-encoding which consist of having a vector for each word in the vocabulary, the vectors are of size equal to the vocabulary size. This vector have all entries at 0, except for one which is the position representing the word, which is set to 1.

There are various methods to induce more meaningful word representation, the two main types are count based and prediction based approaches (Baroni et al., 2014), or following Turian et al. (2010) distributional and distributed representation respectively.

- Count based: are based on co-occurrence context and on the distributional hypothesis: "*linguistic items with similar distributions have similar meanings*", hence the similarity is expressed in terms of the similarity of the distribution. Traditional solutions include the creation of a sparse matrix containing the association of the word with a context. An example of association is the Pointwise Mutual Information (PMI). The problem with this sparse representation is that become computational inefficient, solution like Singular Value Decomposition help with the reduction of these high dimensional and sparse matrices. An important example of count based approach is GloVe (Pennington et al., 2014) which will be described in the next paragraphs.
- Prediction based: the model learn representations by trying to predict the token given the context, or vice versa. Some examples include the work of Collobert and Weston (2008) which for the first time demonstrated the utility of distributed representation for downstream tasks and their proposed neural network architecture forms the foundation for many current approaches, and the Word2Vec framework (Mikolov et al., 2013a,c) which popularized word embeddings.

Levy et al. (2015) shows that both count and prediction methods have the same performance, without one overtaking the other.

In the work "*Word representations: A simple and general method for semi-supervised learning*", Turian et al. (2010) define as "*word embeddings*" only the distributed representation (prediction based), but over time this distinction is not enforced and it is common to refer to all vector word representations as word embeddings.

Another essential part of NLP is the study of language models. A language model is a statistical model of language usage. It consists mainly of predicting the next word given some previous words. A language model learns the probability of word occurrence based on examples of text. Simpler models may look at a context of a short sequence of words, whereas larger models may work at the level of sentences or paragraphs.

More formally, the goal of a statistical language model is to learn the probability $P(x_1, \ldots, x_m)$ for a sequence of tokens $x_1, \ldots, x_m$. We can compute this via the chain rule of probability:

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i | w_1, \ldots, w_{i-1}) \tag{2.14}$$

Because the number of words that precede a word varies, and because it is difficult to compute $P(w_i|w_1, \ldots, w_{i-1})$ with i large, we typically condition the probability of a word on a window of n previous words (n-grams):

$$P(w_1, \ldots, w_m) \approx \prod_{i=1}^{m} P(w_i | w_{i-n}, \ldots, w_{i-1}) \tag{2.15}$$

Unfortunately, this approach doesn't generalize to new sequences, unless using some tricks like Katz (1987) did where they compute small n-grams, generally $n = 3$, and then generalizes to unseen sequences by producing new ones using overlapping n-grams of length up to n. However, this solution does not generalize to sequences longer than n, which has to be small, and also does not take in consideration the similarities of words. Moreover, statistical language models suffer from the problem known as curse of dimensionality that appears when the vocabulary size increases. For example, computing the joint probability of 10 consecutive words having a vocabulary size of 100k there are $100000^{10} - 1 = 10^{50} - 1$ free parameters.

Bengio et al. (2000) developed a solution for the course of dimensionality of statistical language models by proposing the first large-scale language model based on neural networks, referred to as Neural Network Language Models (NNLMs). In their work Bengio et al. (2000) bring together language modeling and word embeddings by 1) associating each word in the vocabulary with a distributed word vector called "*feature vector*", 2) express the joint probability function of word sequences in terms of the feature vectors of these words in the sequence; and 3) learn simultaneously the word feature vector and the parameters of the probability function. This new representation also generalizes to longer sequences and is able to learn similarities between words. Bengio et al. (2000) conclude their work by suggesting the use of Recurrent Neural Networks (RNN) to take advantage of temporal structures and also consider longer sequences like entire paragraphs.

A few years later, Mikolov et al. (2010) propose a neural language model based on RNN. They use a vanilla RNN (Elman, 1990) to show how language models based on recurrent networks are significantly better that previous solution. They perform experiments on various speech recognition task, and achieve better results even when training using lower amount of data.

This work and the one of Bengio et al. (2000) were followed by many other word embeddings solutions that build upon their methods, in particular current state-of-the-art solutions use RNN, or improved versions like Long Short Term Memory networks (Hochreiter and Schmidhuber, 1997). There are two types of

embeddings, the first one is general embeddings which are not dependent on the context like Word2Vec (Mikolov et al., 2013a,c), GloVe (Pennington et al., 2014), and fastText (Bojanowski et al., 2017); the second is context dependent embeddings, these are the new state-of-the-art solution and include ELMo (Peters et al., 2018), and BERT (Devlin et al., 2019). We will present fastText and BERT in Section 5.3 as we compare their multilingual versions for the zero-shot relation extraction task. Now, we present Word2Vec, GloVe, and ELMo.

Word2Vec Mikolov et al. (2013a) proposes two log-linear models, the skip-gram (SG), and the Continuous Bag-of-Words (CBOW). Skip-gram model trains a network to predict the word given the context in which the word appears. While the CBOW does the inverse, it predicts the context of a word given that word. The network is a feedforword neural network with one hidden unit. We have an input layer which is projected onto a hidden layer using a matrix W, the word embeddings matrix, then the hidden layer is projected to the output layer using a matrix $\tilde{W}$, which is the word context embeddings.

Given a corpus $\mathcal{C}$, a vocabulary V, a context of size n, and a word x_t, we define the skip-gram as follows:

$$J_{\text{SG}} = -\frac{1}{|\mathcal{C}|}\sum_{t=1}^{|\mathcal{C}|}\sum_{-n\leq j\leq n, j\neq 0}\log P\left(x_{t+j}|x_t\right) \tag{2.16}$$

while the CBOW:

$$J_{\text{CBOW}} = -\frac{1}{|\mathcal{C}|}\sum_{t=1}^{|\mathcal{C}|}\log P\left(x_t|x_{t-n},\ldots,x_{t-1},x_{t+1},\ldots,x_{t+n}\right) \tag{2.17}$$

Both probabilities are computed using the softmax function:

$$\frac{\exp\left(\tilde{\mathbf{w}}_z^\top \mathbf{w}_s\right)}{\sum_{j=1}^{|V|}\exp\left(\tilde{\mathbf{w}}_j^\top \mathbf{w}_s\right)} \tag{2.18}$$

where $\mathbf{w_i} \in W$, $\tilde{\mathbf{w}}_\mathbf{i} \in \tilde{W}$ are the word and context embeddings for word x_i and can be extracted by multipling the one-hot-encoding of word x_i with matrix W and $\tilde{W}$ respectively. For the SG $\mathbf{w}_s = \mathbf{w}_t, \mathbf{w}_z = \mathbf{w}_{t+1}$, and for the CBOW $\mathbf{w}_s = \sum_{-n\leq j\leq n, j\neq 0}\mathbf{w}_{t+j}, \mathbf{w}_z = \mathbf{w}_t$. This is very inefficient, since for each prediction we need to iterate over the entire vocabulary. An improved version of the softmax is the hierarchical softmax (Morin and Bengio, 2005), which uses a tree structure to reuse already computed probabilities, bringing the complexity to $O(log(|V|))$, with the softmax complexity equals to $O(|V|)$.

In a following work, Mikolov et al. (2013c) propose an efficient way to train NNLMs, called negative sampling, a simplified version of noise contrastive estimation (Gutmann and Hyvärinen, 2012). Negative sampling trains the model to distinguish a target word w_{t+1} from negative samples drawn from a noise distribution P_z. Negative sampling is defined as:

$$P\left(x_{t+j}|x_t\right) = \log\sigma\left(\tilde{\mathbf{w}}_{t+j}^{\top}\mathbf{w}_t\right) + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_z}\log\sigma\left(-\tilde{\mathbf{w}}_i^{\top}\mathbf{w}_t\right) \tag{2.19}$$

where σ is the sigmoid function and k is the number of negative samples. This can be easily adapted also for the CBOW.

Levy and Goldberg (2014) show how SG with negative sampling is an implicit factorization of the Pointwise Mutual Information matrix, which is a count based method.

GloVe Global Vector (Pennington et al., 2014) is an embedding method based on co-occurrence matrix factorization. GloVe use a co-occurrence matrix C, where an entry C_{ij} is the number of times word x_i appears in the context of word x_j. They use it to compute a probability matrix P of a word of being in the context of another word, where P_{ij} is the probability of word x_i being in the context of word w_j. The main idea is that word meanings are captured by the ratios of co-occurrence probabilities rather than the probabilities themselves, they want to model a function F that is able to approximate this ratio using the word vectors, and context word vectors.

$$F\left(\mathbf{w}_i, \mathbf{w}_j, \tilde{\mathbf{w}}_k\right) = \frac{P_{ik}}{P_{jk}} \tag{2.20}$$

To preserve the information of the probability ratio, and maintain the linear structure, the previous equation becomes:

$$F\left(\left(\mathbf{w}_i - \mathbf{w}_j\right)^T\tilde{\mathbf{w}}_k\right) = \frac{P_{ik}}{P_{jk}} \tag{2.21}$$

They find that the best candidate for F is the *exp* function, and via various step, they obtain the following error function:

$$\mathbf{w}_i^T\tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log C_{ij} \tag{2.22}$$

where the b_i and $\tilde{b}_j$ are bias terms. The only problem with this equation is word with low frequency are treated in the same way as the one with high frequency, to fix this they add an weight function f with some conditions. The final cost function is:

$$J_{GloVe} = \sum_{i,j=1}^{|V|} f\left(C_{ij}\right)\left(\mathbf{w}_i^T \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log C_{ij}\right)^2 \tag{2.23}$$

ELMo Proposed by Peters et al. (2018), it is a deep contextualized word representation model. Instead of having a fixed embedding that represents a word, contextualized embeddings are a function of the input sequence and are able to model information like polysemy.

ELMo uses a bidirectional language model (BiLM) on top of which a task specific representation is created. The contextualized representation is a function of the hidden states of the BiLM. Given a BiLM with L layers, it computes a set of $2L + 1$ representations for each token t_k:

$$R_k = \left\{\mathbf{h}_{kj}^{LM}, |j = 0, \ldots, L\right\} \tag{2.24}$$

where $h_{k0}^{LM} = x_k^{LM}$ (the token representation from the language model), and $h_{kj} = [\overrightarrow{\mathbf{h}}_{kj}^{LM}, \overleftarrow{\mathbf{h}}_{kj}^{LM}]$. Then a task specific representation is computed.

$$\mathbf{ELMo}_k^{task} = E\left(R_k; \Theta^{task}\right) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM} \tag{2.25}$$

where E is a function for creating task specific embeddings parametrized by Θ. An generic example is the weighted sum. The weights s^{task} in the linear combination are learned for each end task and normalized by softmax. The scaling factor γ^{task} is used to correct the misalignment between the distribution of the BiLM hidden states and the distribution of task specific representations.

2.4 Transfer Learning

In this section we introduce the definition and the types of transfer learning based on the survey of Pan and Yang (2010), and the thesis of Ruder (2019) in which he applies transfer learning techniques to NLP. We start with the general definition of transfer learning, present an overview of the different types, and then focus our attention on the techniques we use in this work.

2.4.1 Introduction

In the traditional supervised scenario, the availability of training data is not always guaranteed; in some cases, the number of accessible data of the target domain

is scarce, or entirely lacking. Deep neural networks suffer from the unavailability of data even more opposed to more traditional machine learning algorithms. Transfer learning relaxes the hypothesis that training data and test data must be independent and identically distributed (i.i.d assumption), allowing the use of data of a *source domain* and a task on this data, known as *source task*. The knowledge from the source domain, and source task is used as a background knowledge when solving the *target task* from a *target domain*.

Formally, a domain $\mathcal{D}$ is made by two components $\mathcal{D} = \{\mathcal{X}, P(X)\}$, where $\mathcal{X}$ is a feature space, $P(X)$ is a marginal probability distribution over a feature space, and $X = \{x_1, \ldots, _n\} \in \mathcal{X}$. Given a domain, a task $\mathcal{T}$ consists of a label space $\mathcal{Y}$, a prior distribution $P(Y)$, a conditional probability distribution $P(Y|X)$.

Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target conditional probability $P_T(Y_T|X_T)$ in $\mathcal{D}_T$ using the knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.

Since the domains and tasks are tuples, the inequality conditions in the transfer learning definition allows for five different scenarios grouped in two major settings:

a - Inductive learning: the target task is different from the source task, while the source and target domains may or may not be the same. In his setting labeled data in the target domain are required to induce a model.

- $P_S(Y_S) \neq P_T(Y_T)$. The prior distributions are different.
- $P_S(Y_S|X_S) \neq P_T(Y_T, X_T)$. The conditional probability distributions are different, this can be solved with approaches like over-sampling, or SMOTE.
- $\mathcal{Y}_S \neq \mathcal{Y}_T$. The target task has different labels. Depending whether the tasks are learned simultaneously or sequentially we have multi-task learning (MTL) and sequential transfer learning respectively.

b - Transductive learning: the source and target tasks are the same, while the source and target domains are different. In this setting, no labeled data in the target domain are available while a lot of labeled data in the source domain are available.

- $P_S(X_S) \neq P_T(X_T)$. Known as domain adaptation, the marginal probability of the source and the target are different. This scenario is common in combination with different prior distributions and also different task labels.
- $\mathcal{X}_S \neq \mathcal{X}_T$ The feature space between the target and source are different. In NLP, this scenario is referred to as cross-lingual learning or cross-lingual adaptation.

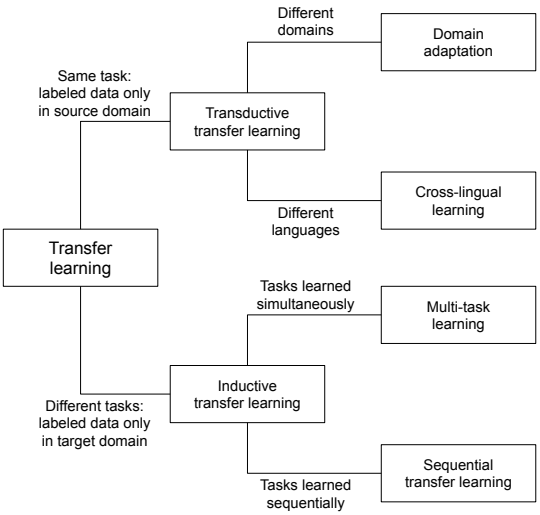


Figure 2.6: Taxonomy for transfer learning in NLP, as in (Ruder, 2019)

Ruder (2019) proposes changes at the taxonomy proposed in Pan and Yang (2010) by adapting the general transfer learning types to the NLP domain. We can see the adapted taxonomy in Figure 2.6.

2.4.2 Multi-task Learning

Traditional machine learning problems rely on a single model, which introduce the risk of ignoring critical source of inductive bias. Caruana (1993) proposes a new learning paradigm based on the idea that related tasks can be used as sources of mutual inductive bias. By introducing this bias, the learning could be faster and more accurate. This approach is called multi-task learning (MTL), or joint learning.

Given a multi-task problem consisting of m tasks $X = \{T_1, \ldots, T_m\}$. For task T_i, its training dataset contains n_i examples $X_i = \{x_{i,1}, \ldots, x_{i,n_i}\}$ as well as their labels $Y_i = \{y_{i,1}, \ldots, y_{i,n_i}\}$. The learning function for task T_i is defined as $f_i(x, w_i)$, and $W = \{w_1, \ldots, w_n\}$. The function to optimize all the task is:

$$\min_{W} \sum_{i=1}^{m} \frac{1}{n_i} \sum_{j=1}^{n_i} \mathcal{L}\left(y_{i,j}, f_i(x_{i,j}, w_i)\right) + \lambda \Omega_W \tag{2.26}$$

this means that the goal of the model is to jointly optimize all the tasks, with all of them having the same importance. This could be changed in case we have a main task and the others are auxiliary task, used to help the main model achieve better generalization.

We present some general works on deep MTL architectures and examples of applications in natural language processing. A type of architecture for MTL consists of a network where the first n layers are shared across all task, and the final layers are task specific. The hard parameter sharing architecture is simple to implement and helps with the reduction of overfitting, but it is only guaranteed to work with closely related task. Another method is the soft parameter sharing, in this case each task has his own layers, and the parameters are then regularized in order to make them similar.

A more general framework that allows the network to learn what and how much to share across all task are the sluice networks (Ruder et al., 2017). In a sluice network each hidden layer is split in m, with m equal to the number of tasks. In a two task setup, we have that each layer is divided in two, one part for the task specific and the other for the sharing of parameters. The communications with the next layer for each task and across tasks is regulated by a two by two matrix with learnable weights. The matrix decides how much information from one task pass to the next layer of the same task, and how much to the next layer of the other task. This reduces the likely of negative transfer.

The use of MTL brings various improvements also in NLP, for example Collobert and Weston (2008) use hard parameter sharing and train a single network on various low level NLP tasks like Part-of-Speech tagging, showing how the general performance of all tasks improves. Moreover, Abdou et al. (2018) compares different types of parameter sharing techniques, from the hard parameter to the sluice networks showing how semantic parsing task helps other syntactic tasks.

2.4.3 Sequential Transfer Learning

Sequential transfer learning is similar to MTL, but instead of having the task learned jointly, we first train one and then the next and so on. This is useful in three scenarios: a) when the data for the tasks are available at different time, b) the source task has much more than the target and c) we need to adapt to various tasks.

There are different techniques to perform sequential transfer learning, but we focus on the fine-tuning methods as we use it for our experiments.

Fine-tuning

Fine-tuning is a method of adaptation of trained models by updating pretrained models using new data. It is more expensive than other sequential transfer solution like feature extraction, but it requires very little task-specific changes and allows the reuse of more general task to model more specific ones. Some solution for fine-tuning consists in training a network and then reuse the first n layers for

another task by retraining them allowing the gradient to propagate through some or all the layers. This solution is not very effective in all tasks since the model could start forgetting due to the high gradients backpropagating at the beginning of the training phase for the new task. One solution is to gradually unfreeze the layers starting from the higher ones going to the first ones, allowing the gradient to change the task specific layer more drastically than the more general.

In (Howard and Ruder, 2018) they propose a method for Universal Language Model Fine-Tuning (ULMFiT) for text classification. Fine-tuning language models has been difficult, due to issues with the required amount of data, and problems with overfitting on smaller dataset or forgetting. ULMFiT solves these problems by proposing novel solutions on like discriminative fine tuning, which manages how the learning rate is set across different layers, making general layers update less than more specific ones. Moreover, they propose a slanted triangular learning rate, that changes with the number of iteration by first rapidly increase until a specific threshold, at which it starts to slowly decrease. They also use the gradual unfreezing. All these solution allow to train a classifier by just using hundreds of in domain documents.

2.4.4 Cross-lingual Learning

Cross-lingual learning aims at extracting information from data in a label-scarce target language by exploiting labeled data from a label-rich language. This is based on the fact that related languages influences each other, meaning that we can extract common information from the corpora of rich languages, and use this information for low-resource ones. The fundamental challenge of cross-lingual learning stems from a lack of overlap between the feature spaces of source language data and that of target language data, and the availability of parallel resources.

In the past years, the number of multilingual works has increased, creating more resource and showing increases in performance for various. An example is XNLI, a cross-lingual natural language inference dataset proposed by Conneau et al. (2018) that is used to evaluate textual entailment models. Parallel resources also allow for better multilingual representation, that are effective features for cross-lingual models.

Most of the work on multilingual representation has been on word level, on which we focus in the next section. However, there are also sentence level representations, such as in (Conneau et al., 2018) where they train the target language encoder to represent sentences in the same space as the source language encoder, allowing the transfer. Similarly, Artetxe and Schwenk (2018) train an encoder to represent sentences in a space that is agnostic from the language, however, they pretrain the decoder on parallel corpora and concatenate a language ID to the decoder input that specifies the language to generate the output for.

Cross-lingual Word Representation

Cross-lingual word representation are a useful way to transfer knowledge across languages by providing a common representation for word in different languages. This usually requires some amount of supervision, like small bilingual lexicon, but in the recent years solutions with no supervision appeared (Lample et al., 2018).

An interesting count based solution that does not require parallel corpora was proposed by Søgaard et al. (2015), they use Wikipedia as their multilingual corpora and, based on the idea that the same word translated appears in the same Wikipedia articles in the corresponding language, they build an inverted index matrix by taking the articles common across all languages the embeddings will be created. Finally, they perform dimensionality reduction on the inverted indexing matrix.

Another solution consists in aligning the vector spaces of the languages. Word alignments are generally modeled as a linear mapping, which are constrained such that the structure of the initial monolingual spaces is left unchanged. When having a parallel lexicon, one solution (Mikolov et al., 2013b) to find a linear transformation can be minimizing:

$$\begin{aligned} W^* &= \underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} ||WX - Y||_{\mathbb{F}} = UV^\intercal \\ \text{with, } U\Sigma V^\intercal &= SVD(YX^\intercal) \end{aligned} \tag{2.27}$$

where d is the size of the embedding, X and Y are the embeddings of the lexicon, one in the source language an the other in the target. The matrix W is an orthogonal matrix of size $d \times d$, this has been shown to lead to better performances (Xing et al., 2015), and $||\cdot||_{\mathbb{F}}$ is the Frobenius norm. The translation t of any source word s is defined as $t = \text{argmax}_t \, cos(Wx_s, y_t)$.

2.5 Zero-shot Learning

In this section, we present an overview of zero-shot learning. It studies how machine learning models could generalize to unseen categories. In recent years, the number of zero-shot learning methods proposed every year has been increasing rapidly. We give a description, discuss some general techniques, and show applications in natural language processing tasks.

2.5.1 Introduction

Classification is one of the main tasks in machine learning; it is straightforward: given an example consisting of various feature, assign a label to the example. Thus, it has been thoroughly studied in different domains and tasks, from computer vision to natural language processing. The main issue with this setup is that it does not generalize to new labels. Zero-shot learning (Larochelle et al., 2008) can be seen as an extreme version of few-shot learning, and one-shot learning (Miller, 2002).

In their survey, Xian et al. (2017) define the zero-shot learning task as follows: Given a training set $\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, with $y_i \in Y^{tr}$ the training set classes, the task is to learn a function $f : X \to Y$ that minimizes the loss function $J(\theta)$, and F is a compatibility function that gives how compatible the input and the label are:

$$\begin{aligned} \min_{\theta} J(\theta) &= -\frac{1}{N}\sum_{1}^{N} \mathcal{L}(y_i, f(x, \theta)) + \lambda\Omega_\theta \\ f &= \underset{y \in Y}{\operatorname{argmax}}\, F(x, y; \theta) \end{aligned} \tag{2.28}$$

At test time, the aim is to assign to new examples an unseen class label, or in the generalized zero-shot, an unseen or a seen class label. This is achieved by providing the model with a description of each class in some representation, the simplest being a vector of numeric or symbolic attributes.

Akata et al. (2016) embeds each class in the space of attribute vectors, their method is called Attribute Label Embedding (ALE). They propose the use of label embeddings as the class description, and train a compatibility function F that minimizes the distance between the label and the image representation. Their F function is:

$$F(x, y; \theta) = \gamma(x)^T \theta \phi(y) \tag{2.29}$$

where γ is the input embedding, and ϕ the label embedding (e.g word embedding of the image label).

2.5.2 Zero-shot in NLP

Many works in NLP use zero-shot learning methods for different tasks, for example Obamuyide and Vlachos (2018) perform zero-shot classification using a textual entailment framework called ESIM (Chen et al., 2017). Textual entailment consists of predicting if a sentence, the premise, entails, contradicts or is neutral to another sentence, the hypothesis. By reframing relation classification as a textual entailment task, the text to check if the relation exists is the premise, and the hypothesis is the description of the relation. ESIM is a three-step framework consisting of an input encoding using BiLSTM, a local inference modelling based on attention and vector composition, and the third layer is the inference composition again using the BiLSTM. Obamuyide and Vlachos (2018) change the ESIM framework, by using a conditional BiLSTM (Rocktäschel et al., 2016) (cBiLSTM). The cBiLSTM condition the encoding by using the hypothesis encoding as initialization. This allows the model to perform zero-shot classification as the relation information is used in the encoding of the premise.

Another example is for the machine translation task. In (Johnson et al., 2017) they use an encoder-decoder architecture with 8 layer LSTMs and attention mechanism. They enable zero-shot translation by prepending the example strings in the source language with an extra token for the target language and having a shared encoding space for all languages. The same solution is also used in (Bollmann et al., 2019) for historical text normalization, with an additional token for the task as they also have an auxiliary task in a multi-task learning setup.

Chapter 3

State-of-the-Art

In this Chapter we present the literature for the relation extraction task (Section 3.1) and for the machine reading comprehension (Section 3.2). Each section gives a definition of the task, and present an overview of the solutions adopted to solve the task. We also show how techniques are adopted in a multilingual setup, and for the relation extraction task, present some literature on the zero-shot scenario.

3.1 Relation Extraction

In the following sections we present the relation extraction task. We start with a definition, and present some approaches to solve the task. We present works for both multilingual and zero-shot learning scenarios.

3.1.1 Introduction

Many applications in information extraction, natural language understanding, information retrieval require an understanding of the semantic relations between entities. This information is usually contained in unstructured data like blogs, news articles, and so on. Relation extraction aims at making this information usable by other system by extracting the relations contained in unstructured documents and make is available in knowledge bases.

A relation is defined in the form of $r(e_1, e_2, ..., e_n)$ where the e_i are entities in a relation r within document D. We focus on relations with only 2 entities, and refer to the entity pairs and relation as triples, as is common also the use of the following notation (e_1, r, e_2). Since our dataset is built from Wikipedia and Wikidata, throughout this work, we use the term property interchangeably with relation. Given human readable text, relation extraction is the task of identifying

instances of relations $r(e_1, e_2)$. For example, in the sentence: "Elon Musk is the CEO of both Tesla and SpaceX", a system should be able to extract the relations *chief_executive_officer(Elon Musk, Tesla)*, and *chief_executive_officer(Elon Musk, SpaceX)*.

Simple relation extraction can be achieved using rules, however, rule-based systems require a deep knowledge of the domain and high human annotation time. In the supervised setting (Bach and Badaskar, 2007), depending on the nature of input to the classifier, approaches are further divided into feature based methods and kernel methods. For the feature based systems both the semantic and syntactic features are used as input to the classifier in the form of a feature vector. In (Kambhatla, 2004) they use maximum entropy models, or (Zhao and Grishman, 2005; Zhou et al., 2005) uses SVMs trained on the features with different kernel types.

These solution however still rely on human domain knowledge for the definition of the features, this can be solved using end-to-end solutions like in (Miwa and Bansal, 2016; Pawar et al., 2017). In an end-to-end relation extraction the system has to identifies both the entities, their type and the relation holding between them, if any. Miwa and Bansal (2016) uses a biLSTM on both the word sequences, and tree structures. In Pawar et al. (2017) they train a joint model to predict the boundaries, the type of the entities and the relation between them.

The clear drawback of supervised methods is the need of training data: data labeling is expensive, and there is often a mismatch between the training data and the data the system will be applied to. A solution to this problem has been proposed in (Plank and Moschitti, 2013). They use domain adaptation techniques, and build a model that uses syntactic tree kernels enriched by lexical semantic.

An alternative type of relation extraction is Open Relation Extraction (Open RE) (Banko et al., 2007; Banko and Etzioni, 2008). Instead of having a fixed set of relations, Open RE aims at extracting relational triples from an open-domain corpus. In (Banko et al., 2007) in addition to the introduction of the Open RE task, they propose TextRunner, a large scale relation extraction tool on open domains. TextRunner uses a self-learner to automatically label if there exists a relation between the extracted entities in the text. The examples, both positive and negative are used to train a Naive Bayes classifier. The use of patterns between mentions as relations removes the requirement of supervision and has high flexibility, however it lacks generalization. A solution to the generalization issue is by clustering the surface form of words with similar meaning (Yao et al., 2011).

Riedel et al. (2013) remove the need of a defined schema by using Universal Schemas, the union of all involved schemas (surface form predicate as in (Banko et al., 2007), and relations in the schema of pre-existing databases). They model the problem as a probability of a relation holding between two entities, this allows

them to use solutions from collaborative filtering. In particular they perform matrix factorization on the matrix with entity tuples are rows, and the columns are the union of the surface forms and the DB relations. More formally, they model the probability of two entity $t = (e_1, e_2)$ being in relation r by using an exponential function parametrized by a natural parameter $\theta_{r,t}$:

$$P(y_{r,t} = 1|\theta_{r,t}) = \frac{1}{1 + exp(-\theta_{r,t})} \tag{3.1}$$

the parameter $\theta_{r,t}$ can be computed in different ways (e.g. the the dot product of the latent feature representation of the relation and the tuple).

By defining the matrix using the entity pairs, we can only predict relations between entity seen at train time. Verga and McCallum (2016) propose a row-less universal schema. Instead of encoding each pair explicitly, they use aggregation functions over the over the observed instances. The embedding of the pairs can be seen as a summarization of all relation types the entities were seen in.

3.1.2 Multilingual Relation Extraction

Earlier works investigating multilingual relation extraction have been few and far between. In Faruqui and Kumar (2015) they perform multilingual RE by translating the sentences in English and the run a Open RE system. This solution is simple and relies only on the availability of a translator from the target language to English. Verga et al. (2016) uses universal schemas on top of multilingual embeddings to extract relations in a target language without using training data in the target language.

Lin et al. (2017) proposes one of the first neural multilingual solution for RE. They use use an dataset of English articles from Wikipedia, and Chines articles from Baidu Baike (a Chinese-language online encyclopedia). The articles were aligned and annotated using Wikidata. They use a sequence-to-sequence model with both a cross-lingual and mono-lingual attention. The cross-lingual attention helps with sentences with strong consistency across the languages.

3.1.3 Zero-shot Relation Extraction

Zero-shot relation extraction was introduced by Rocktäschel et al. (2015). The task is the same as for relation extraction, but instead of having a fixed amount of pre-defined relation that are both in the training and test set, the relations at test can be seen or unseen during training. They propose a solution to the task by injecting logic into relation embeddings and then use matrix factorization methods. Levy et al. (2017) proposes a new approach to relation extraction that allows zero-shot learning. They propose a novel approach towards achieving this generalization

by transforming relations into natural language question templates. This means that we can translate a relation *born_in(X, Y)* into the question *"When was X born?"*, or *"In what year was X born?"* . Reframing RE as a question answering problem, allows the use of reading comprehension models. In particular, they use BiDAF (Seo et al., 2017), a machine comprehension model that uses BiLSTM and attention mechanism to find the answer of a question given a context (more detailed description in Section 3.2.2). The reading comprehension model is trained using the question, answer, and context examples where the X slot is filled with an entity and the Y slot is either an answer, if the answer is present in the context, or `NIL`. The model is then able to extract relation instances from raw text, even for relations which has not been seen during training, enabling zero-shot relation extraction.

Another solution has been proposed by Goldstein (2018). They achieve zero-shot relation extraction using word embeddings. They construct triples (A, r, B), where A, B are matrices where each column vector represents an entity. Entities in the same column but in the other matrix are in relation r. The triples are used as a starting point to find other entities that are in the same relation r. A, B are obtained by performing bootstrapping using a seed entity on the word embedding vector space. After the small set of entities was found, they are used to generalize to new ones. This is achieved by finding the most common features between all possible pairs on entities, and reducing the space to only these features enabling to find other entities in relation using the distance of the points. A method based on embedding vector space brings different advantages like the ability to find duplicate relation (e.g. misspelled relations), and the ability to correct the model by using the new added information.

3.2 Machine Reading Comprehension

In the next sections we present the Machine Reading Comprehension task. We introduce the literature and take a closer look at a model, to better understand how these models work.

3.2.1 Introduction

Childrens' ability to read is evaluated by giving them reading comprehension tests. These test typically consist of a short paragraph followed by questions. The questions are designed in such a way that to answer correctly the reader has to understand the content of the text. Similar tests are also used to measure machine's ability to understand documents. Machine Reading Comprehension (MRC) has

gained a lot of interest in the last years with the introduction of sequence-to-sequence models, and datasets like SQuAD (Rajpurkar et al., 2016, 2018).

The reading comprehension task easily conforms to a formulation as a supervised learning problem. Specifically, given a context document c, a query q relating to that document, and the answer a to that query, goal is to estimate the conditional probability $P(a|c, q)$. Depending on the type of the question an the answer, reading comprehension can be of four types:

- Cloze-style: same as in the Cloze task (Taylor, 1953), the question contains placeholder that the system has to fill;
- Multi-choice: the answer can be only one of multiple choices;
- Span-prediction: the machine needs to find the correct answer in the passage by predicting the spans. In this case, in newer datasets, it is also included the possibility of no answer (`NIL` answer).
- Free-form: the answer is human-generated, so there are some problems with the evaluation.

Traditional systems rely on solutions from information extraction, by detecting predicate argument triples that can later be queried as a relational database (Poon et al., 2010) or pattern matching, and use statistical and mathematical methods to perform similarity between the question and the answer to find the correct answer. For example, in (Riloff and Thelen, 2000) they propose QUARC (QUestion Answering for Reading Comprehension) a rule-based system that uses lexical and semantic heuristics to look for an indication that a sentence contains the answer to a question. Each type of *"Wh"* question looks for different types of answers, so QUARC uses different sets of rules for each question type. These solutions are not actually understanding a given passage, so they lack the reasoning we would expect from a reading comprehension model.

Recent solution are based on neural networks, and popular architectures like the sequence-to-sequence model. These methods all share a common structure as described in (Qiu et al., 2019). The passage and the answer both pass to an embed layer,an encoding one, then the encodings are used together by the match layer, which outputs to the answer layer. The layers role are:

- Embed layer: this layer converts the words into their representations as seen in Section 2.3;
- Encoding layer: produces a representation of the input, this can be produced using LSTMs, or other sequence encoding models;
- Match layer: captures the interaction between the context and the question, and produces a query-aware context representation;
- Answer layer: it predicts the correct answer, the answer can be of different types as mentioned before.

The use of neural networks solutions, and sequence-to-sequence models for reading comprehension was popularized by (Hermann et al., 2015; Weston et al., 2015). Weston et al. (2015) proposes a new model architecture called Memory Network. Memory Network, which is a framework with four components 1) a component that transforms the input into the internal representation, 2) one that changes the memory states using the input, and the memory, 3) the component that generates the the output features based on the memory, and the input and 4) the last component that decodes the features to predict the output. The component can be any model (SVM, NN, etc.). They also propose a Memory Neural Network (MemNN) for the reading comprehension task.

Hermann et al. (2015) uses RNNs with attention mechanism, and proposes two models, the Attentive Reader and the Impatient Reader. The first one encodes the query using bidirectional LSTM and concatenates the results to get the representation of the query. The representation of the document is formed by a weighted sum of the BiLSTM outputs at each time step. The model is completed with the definition of the joint document and query embedding using a nonlinear combination. The Impatient Reader improves the representation of the query by using a the same method employed for the document encoding.

The two architectures proposed in (Hermann et al., 2015) are at the core of many other solutions (Kadlec et al., 2016; Chen et al., 2016). In (Cheng et al., 2016), similarly to (Weston et al., 2015), they increase the memory of the RNN. In particular they use a LSTM with a memory network instead of a single cell, the memory network enables adaptive memory usage and offers a way to weakly induce relations among tokens. There are also multilingual reading comprehension solutions. Asai et al. (2018) uses a pre-trained reading comprehension model in English in combination with a neural machine translation system that converts the context and the query from the target language into English, and does the same for the answer.

3.2.2 BiDAF

Levy et al. (2017) uses as their machine comprehension model BiDAF (Seo et al., 2017). BiDAF (Bi-Directional Attention Flow) instead of using a single attention and/or focusing only on small portion of the text like previous solution did, it uses a multi-stage hierarchical process that represents the context and query at different levels of granularity and uses bidirectional attention flow mechanism to obtain a query-aware context representation. It follows the general architecture of all reading comprehensions model.

Embed Layer

Character embedding layer is responsible for mapping each word to a high dimensional vector space. BiDAF uses both character and word level embedding. Given a context $c = \{c_1, \ldots, c_T\}$ and a query $q = \{q_1, \ldots, q_J\}$, they are separately encoded using the concatenation of the character and word embedding vectors which are passed to a two layer Highway LSTMs (Srivastava et al., 2015). The output are the representations of the context $C \in \mathbb{R}^{d \times M}$, and query $Q \in \mathbb{R}^{d \times M}$ at each step. These are passed to the encoding layer.

Encoding Layer

The encoding layer uses a BiLSTM to model the temporal interaction between words. The embeddings from the previous step are further encoded into $H \in \mathbb{R}^{2d \times N}$ for the context, and $U \in \mathbb{R}^{2d \times M}$ for the query. The next step is to find the interaction between them.

Match Layer

BiDAF match layer is further divided in two, the attention flow layer, and the modeling layer. The former is responsible for finding the relevant information in the context regarding the query, and the vice versa. This is achieved by linking their information using attention. They compute two attention, a context-to-query and a query-to-context, they are a function of similarity matrix $S \in \mathbb{R}^{N \times M}$:

$$S_{ij} = \alpha(H_{:u}, U_{:j}) \tag{3.2}$$

where α is a trainable scalar function that encodes the similarity between the vectors $H_{i:}$, and $U_{i:}$. The context-to-query attention is computed using the the attention weights $a_t = softmax(S_{t:}) \in \mathbb{R}^J$ on the query words by t-th context word. The final attention matrix is $\tilde{U}$ contains the attended query vectors for the entire context, and $\tilde{U}_{:t} = \sum_j a_{tj} U_{:j}$.

In the query-to-context attention the weights are $b_t = softmax(max_{col}(S) \in \mathbb{R}^T$, and the attended context vector $\tilde{h} = \sum_t b_t H_{:t} \in \mathbb{R}^{2d}$ is the weighted sum of the most important words in the context. The final matrix $\tilde{H} \in \mathbb{R}^{2d \times T}$ is a T times stack of $\tilde{h}$.

Instead of using the attention to summarize the information and risk to loose some of it due to the early stage of the operation, the computed context-to-query, and query-to-context attention together with the embeddings from the encoding layer are used to create a query-aware representation of each context word G, where:

$$G_{:t} = \beta(H_{:t}, \tilde{U}_{:t}, \tilde{H}_{:t}) \tag{3.3}$$

where β is any function. Then G is used as an input by the modeling layer, that uses a two-layer BiLSTM to create the final representation M used for predicting the output.

Answer Layer

The answer layer uses a dense neural network with a softmax layer to predict the beginning of the answer in the context, the output of the softmax is also used as an input for an LSTM that predicts the end of the sequence. Levy et al. (2017) modifies the final layer for the possibility of `NIL` answer.

Chapter 4

X-WikiRE

In this chapter we will describe how we built **X-WikiRE**. We will start by describing the datasources used, Wikipedia and Wikidata. Next, we will describe the process we used to combine these resources to obtain automatically annotated data for different languages. As final step, we will describe how to increase the number of examples by creating different templates. In the last section, we will show some descriptive statistics about **X-WikiRE**.

4.1 Introduction

Supervised training of deep neural networks often requires large amounts of high-quality training data which are difficult to obtain. To fix this, Hewlett et al. (2016) exploited Wikidata (Vrandečić and Krötzsch, 2014) and Wikipedia to create large amounts of distantly annotated data for various natural understanding tasks. Levy et al. (2017) use WikiReading as a starting point for building a large scale reading-comprehension based dataset for relation extraction. Based on the aforementioned works we build **X-WikiRE**, a new large scale multilingual reading-comprehension based dataset for relation extraction. **X-WikiRE** is available in English, Spanish, Italian, German, and French, and is also on at least a magnitude order larger than other multilingual resources for RE.

4.2 Datasources

Our datasources are Wikipedia, and Wikidata. Wikidata it is a free collaborative knowledge base that is available in more than 300 languages. Its value has long been obvious, with many efforts to use it. The Wikidata approach is to crowd source data acquisition, allowing a global community to edit its content, making it a good quality resource. Wikipedia is a online encyclopedia with articles available,

as for Wikidata, in more than 300 languages. Wikipedia consists of textual articles describing entities.

We can see Wikidata as a collection of triples `(entity_id, property_id, value)`. Each entity is identified by a unique number, prefixed with the letter Q, known as a "QID". Also properties, which define the relation between the entities, are identified by a prefix P and a number (e.g. "`P35`" is the id for the property *"head_of_state"*). The relations each entity holds with other entities are called *statements*, so a document corresponding to an entity, is a set of triples or statements. Values could be either other entities, or numeric/textual values. For example, given the entity *Niels Bohr*, with id `Q7085`, the relation *born_in* (with id `P19`), and the entity *Copenhagen*, id `Q1748`, we will have the triple `(Q7085, P19, Q1748)` stating that Niels Bohr was born in Copenhagen. Items may be linked to articles on Wikipedia.

4.3 X-WikiRE

The first step to obtain annotated textual data is to combine the Wikidata triples with the corresponding entity's article in Wikipedia. For each entity in Wikipedia we take the corresponding Wikidata document, add the Wikipedia page text, and denormalize the statements. The denormalization consists of replacing the ids in each statement of the document with the text label for entities and values which are entities, and with the human readable form for numeric values. For example, given a statement `(Q7085, P19, Q1748)`, the denormalization replaces the ids with the label, obtaining `(Niels Bohr, born_in, Copenhagen)`. In case of values, for example a date (encoded as a timestamp), the tuple `(Q7085, P569, -2658182400)`, will be converted in the human readable statement is `(Niels Bohr, born_on, 7 October 1885)`.

Now that we have our resources combined, we can extract the annotated sentences. We take the first sentence (context) in the article that contains the entity and the value. This follow the principle that if they are in the same sentence the relation between them is the one expressed in the statement. These are the positive examples. Negatives examples, contexts without the value present, are constructed by finding pairs of triples with common entity type (to ensure they contain good distractions), and swapping their context if the value is not present in the context of the other triple. The swapping allows the creation of good negative examples that contain distractors that can trick the model. The extracted triples, are also uniquely identified with the QID, this mean that we can create a set of parallel examples across languages.

The next step is the querification, that allows to obtain different examples for a single relation by creating templated questions.

4.3.1 Querification

Now that we have the entity, the value and the context, the challenge is to convert the relation and the entity into a question for the machine comprehension model (Table 4.1). Levy et al. (2017) created 1192 questions template for 120 relations. A templated question, is a question with a placeholder for the entity, for example for property *author*, we can have following template *"Who wrote X?"*, where X is the entity placeholder, for example "*Divina Commedia*". So we have that our example are made by a question, $question \approx template(property, x)$), a context, and the answer.

We consider the same 120 properties as Levy et al. (2017), for English we used the templates they created, while for the remaining languages we used human annotators to translate the templates. The annotators were instructed to take a look at the 1192 templates and try to translate them into their language, they were allowed to discard translation that did not make sense, and also create templates inspired by the original templates. In addition to the entity placeholder, some languages with richer morphology (Spanish, Italian, and German) required extra placeholders in the templates because of agreement phenomena (gender). We added a placeholder for definite articles, as well as one for gender-dependent filler words. The gender is automatically inferred from the Wikipedia page statistics and a few heuristics. For example, for Italian, the property *born_in*, will have a template *"Quando è natZ X?"*, where depending on the gender of X, Z can be either *o* or *a*, giving *"nato"* for masculine words, and *"nata"* for feminine ones.

This process created different amount of templates and so the number of examples (see Section 4.4).

The process to create **X-WikiRE** allows to easily construct a similar dataset also for other low resource languages. The only annotation required is the translation of the questions, which can be done using automated translation tools, or turks for a reasonable cost (Levy et al., 2017).

4.4 Dataset Statistics

X-WikiRE covers more languages (five) and is at least an order of magnitude larger than existing multilingual relation extraction datasets, e.g. TAC 2016 (Ellis et al., 2015), which covers three languages and consists of approximately 90K examples. Instead our dataset provides millions of triples, and if we also consider the templates, the amount of examples raises at ten of millions as we can see in Table 4.2.

Lang	Question	Context & Answers
DE	In welchem land befindet man sich, wenn man **Amazonas** besucht?	Der Fluss Amazonas gab seinerseits dem Amazonasbecken sowie mehreren gleichnamigen Verwaltungseinheiten in **Brasilien**, **Venezuela**, **Kolumbien** ...
EN	What country is **Amazon** located in?	The Amazon proper runs mostly through **Brazil** and **Peru**, and is part of the border between ...
ES	¿En qué país se encuentra el **Amazonas**?	El río Amazonas es un río de América del Sur, que atraviesa **Perú**, **Colombia** y **Brasil**.
FR	Dans quel pays peux-tu trouver **Amazone**?	Le fleuve prend alors le nom d'Amazonas au **Pérou** et en **Colombie**, puis celui de rio Solimões en entrant au **Brésil** au ...
IT	Di quale nazione fa parte il **Rio delle Amazzoni**?	Il Rio delle Amazzoni è un fiume dell'America Meridionale che attraversa **Perù**, **Colombia** e **Brasile** ...

Table 4.1: Examples from our dataset of the same question-context pairs across all the languages with the correct answers highlighted in boldface.

Language	Pos	Neg	Pos*	Neg*
DE	2.5M	545K	11M	2.3M
EN	5.1M	1M	64M	12M
ES	1.2M	211K	5.5M	1.1M
FR	2.3M	867K	18M	6.8M
IT	1.9M	217K	10M	1.2M

Table 4.2: The number of positive and negative triples for each language with (*) and without templates.

These examples are distributed differently based on the property and the language. In Figure 4.1, we can see how the different languages have different amount of triples, and also even if English is the language with the highest amount of triples, for some property we have more of them in other languages. An example is the relation *cast_member*, in which French has hundred of thousand more triples. This also means that we have different amount of common triples across languages as we can see in Figure 4.2.

Since we will compare our solution against the one of Levy et al. (2017), we computed the average length of the context in out dataset and their dataset. Observe from Table 4.3 that in our dataset contexts are longer on average. Also observe that, on average, contexts in test set have more tokens than those in the training or development sets for all languages except Italian.

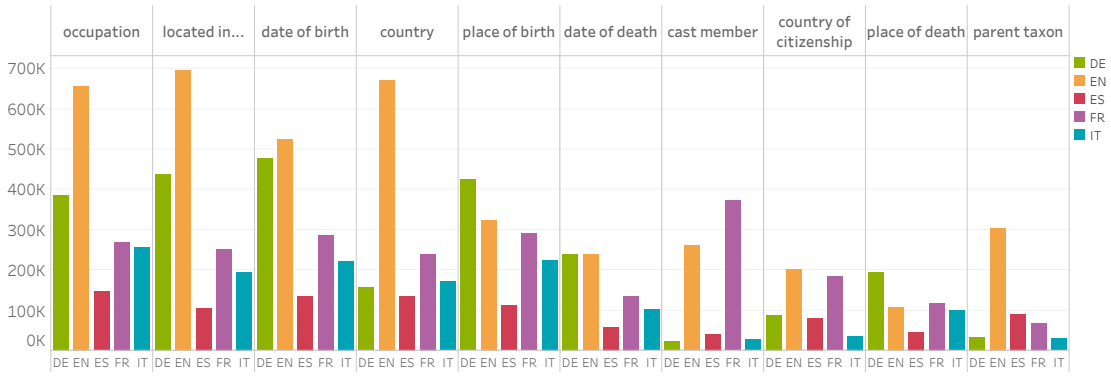


Figure 4.1: The number of triples for the top 10 properties in each language.

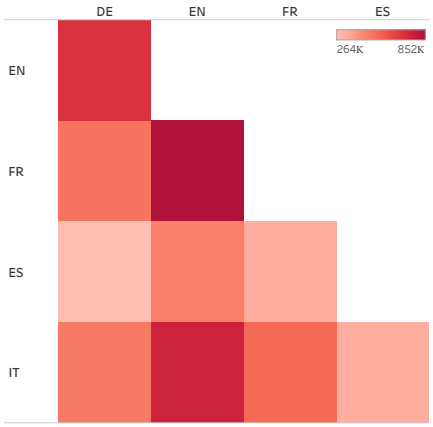


Figure 4.2: The overlap of triples between languages.

	Lang	Train	Dev	Test
Levy et al. (2017)	EN*	29	28	29
X-WikiRE	EN	30	33	34
	ES	34	46	48
	FR	49	39	42
	DE	29	30	32
	IT	40	35	35

Table 4.3: Average number of tokens in the context.

Chapter 5

Proposed Solution

In this Chapter we present the different parts of our setup. We start by introducing the word representation models we used in Section 5.3; next, in Section 5.2 the reading comprehension model and in Section 5.1.1 and 5.1.2 we describe the proposed architectures to benchmark the generalization abilities of the model by performing zero-shot experiments, both using unseen entities and unseen relations.

5.1 Transfer Setups

We use two setups to evaluate the use of a multilingual resource for the relation extraction task. These two solutions share a common nil-aware question answering model, and also the word representation used by the models are multilingual, to allow the transfer. The first is a fine-tuning setup, the second is a multi-task learning. These models will be evaluated on two scenarios to find if the zero-shot relation extraction also benefits from our multilingual resource.

5.1.1 Cross-lingual Model Transfer

This setup (Figure 5.1) is used to see how well RE models can be transferred across languages. In this scenario, the first language has more data, while the target language is a low resource one. The knowledge transfer is aided by the use of a multilingual word representation.

The first step is to pretrain the model on the source language. Then, in the second step, the model is finetuned using the examples in the target language. The fine-tuning is performed on the entire model except for the embeddings which are kept frozen.

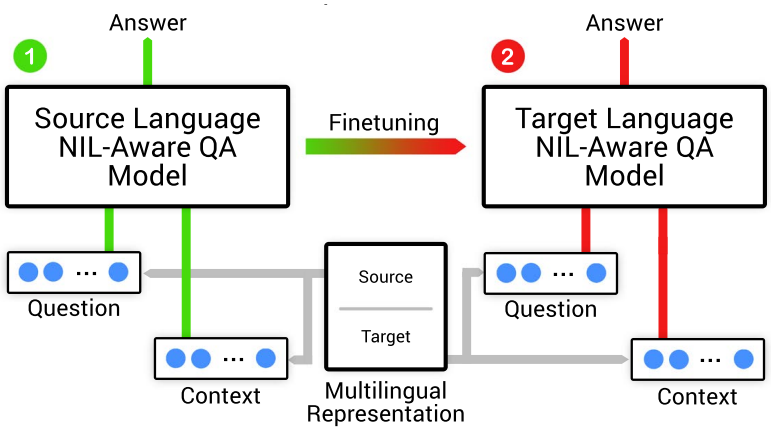


Figure 5.1: Cross-lingual model transfer. In step (1), a source language model is trained until convergence. In step (2), it is finetuned on a limited amount of target language data.

5.1.2 One Model, Multiple Languages

Shown in Figure 5.2, this setup trains a single model using multiple languages. This model test the ability of a single model to perform RE across multiple languages. In this architecture, the network is trained to maximize the objective function of all languages jointly. Beside of the benefits of MTL, the model is also using the signal from other languages to better generalize unseen entities or relation in a specific languages.

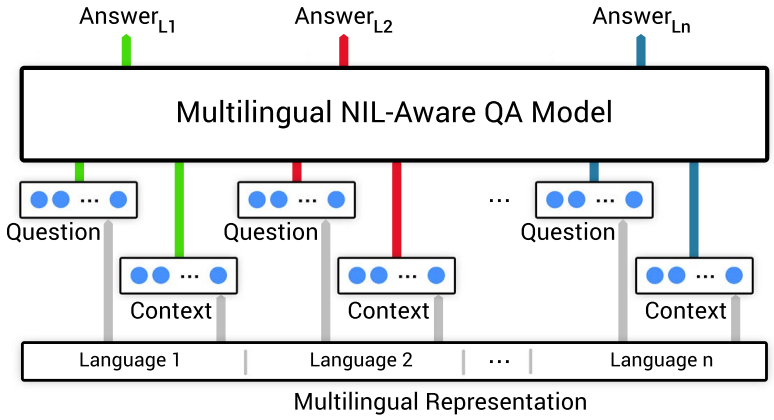


Figure 5.2: Joint multilingual training.

5.2 NIL-Aware QA Model

The core of our multilingual relation extraction setups is the reading comprehension model. In our case we use NAMANDA (Kundu and Ng, 2018a). Previous neural models for reading comprehension based question answering (Seo et al., 2017; Yang et al., 2017) focus on context-question interaction to capture similarities to extract the answer span. However, they don't model the question/answer type and also are unsuccessful to aggregate information from multiple sentences. AMANDA (Kundu and Ng, 2018b) (An end-to-end question-focused Multi-factor Attention Network for Document-based question Answering), and the nil-aware version NAMANDA (Kundu and Ng, 2018a) solve the previous issues by using a multi-factor attentive encoding based on tensor transformation and a max-attentional question aggregation mechanism to learn the meaningful pars of the question.

AMANDA and NAMANDA share the same structure with the only difference in the answer layer, where NAMANDA adds the possibility for unanswerable question. We describe NAMANDA (Figure 5.3) by grouping the different layers into the general structure of reading comprehension models (Qiu et al., 2019).

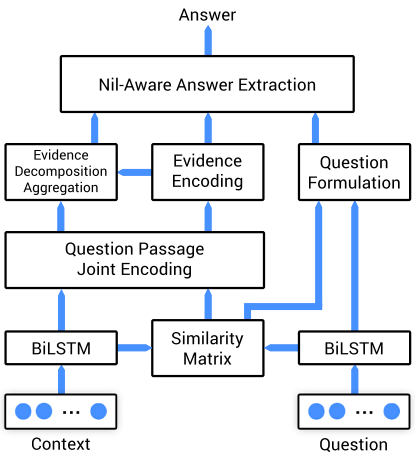


Figure 5.3: An overview of NAMANDA's architecture.

Embed Layer

NAMANDA uses word-level embedding by concatenating the word embeddings from GloVe (Pennington et al., 2014) and a CNN based character embedding. Using character embedding is useful for out-of-vocabulary words. In our experiments we use different embeddings.

Encoding Layer

To encode the contextual information in the context and the question, a BiLSTM is used. Given a context $c = \{c_1, \ldots, c_T\}$ and a query $q = \{q_1, \ldots, q_T\}$, we fed them into two separate LSTMs obtaining $\mathbf{P} \in \mathbb{R}^{T \times d}$, and $\mathbf{Q} \in \mathbb{R}^{U \times d}$, with d the number of hidden units. The rows of these matrix is the concatenation of the forward and the backward hidden state at each time step.

Match Layer

NAMANDA has various layers that help performing the match, these are:

- **Similarity Matrix**: it express the similarity between the words in the passage and the words in the question. The similarity matrix $\mathbf{A} \in \mathbb{R}^{T \times U}$ is defined as $\mathbf{A} = \mathbf{D}\mathbf{Q}^\intercal$.

- **Question Formulation**: the most relevant parts of the question are extracted using a) a max-attentional question aggregation (q_{ma}) and b) a question type representation (q_f). The important parts of the question are aggregated in $q_{ma} = k\mathbf{Q}$, where $k = softmax_{col}(A)$. The question type is encoded as a concatenation of the first *"Wh"* word $q_{t_{wh}}$, and its following word $q_{t_{wh}+1}$ (the t_{wh}-th, and $(t_{wh}+1)$-th rows of $\mathbf{Q}$), $q_f = [q_{t_{wh}}, q_{t_{wh}+1}]$. The final question representation computed using a feed-forward network $\tilde{q} = FF([q_{ma}, q_f]; W_q)$.

- **Question-Passage Joint Encoding**: the encoding is achieved using a row-wise softmax over the similarity matrix $\mathbf{R} = softmax_{row}(\mathbf{A})$. The rows in $\mathbf{R}$ measure how relevant the words in the question are w.r.t the context. The aggregated question representation $\mathbf{G} \in \mathbb{R}^{T \times d}$ is computed as $\mathbf{G} = \mathbf{R}\mathbf{Q}$. The concatenation of the aggregated question vectors and the passage encoding are concatenated in $\mathbf{S} \in \mathbb{R}^{T \times 2d}$, then a BiLSTM is applied obtaining $\mathbf{V} \in \mathbb{R}^{T \times d}$.

- **Evidence Decomposition Aggregation**: The information between multiple sentence is aggregated using a multi-factor attentive encoding using tensor-based transformation. This allows to collect information at different grain level for long contexts. Given the number of factors m, the multi-factor attention $\mathbf{F}^{[1:m]} \in \mathbb{R}^{T \times m \times T}$ is formulated as:

$$\mathbf{F}^{[1:m]} = \mathbf{V}\mathbf{W}_f^{[1:m]}\mathbf{V}^\intercal \tag{5.1}$$

  where $\mathbf{W}_f^{[1:m]} \in \mathbb{R}^{d \times m \times d}$ is a 3-way tensor. The evidence are refined using a max-pooling over the factors, a self-attention matrix $\mathbf{F} \in \mathbb{R}^{T \times T}$ is obtained.

$\mathbf{F}$ is further processed using a row-wise softmax obtaining $\tilde{\mathbf{F}}$. Then the self-attentive encoding is $\mathbf{M} = \tilde{\mathbf{F}}\mathbf{V}$, with $\mathbf{M} \in \mathbb{R}^{T \times H}$. The self-attentive encoding is concatenated with the question-dependent passage, and then a feed-forward neural network-gating is applied, obtaining $\mathbf{Y}$, with $\mathbf{Y} \in \mathbb{R}^{T \times 2d}$.

Following (Wang et al., 2016), the evidence vectors for each context word are decomposed using orthogonal decomposition. Each row y_t of $\mathbf{Y}$ is decomposed with respect to the question-passage joint encoding ($\mathbf{S}$) vector s_t into its parallel components $y_t^{=}$, which represent the relevant parts of the accumulated evidence, and perpendicular components $y_t^{\perp}$ that represent the irrelevant parts.

$$y_t^{=} = \frac{y_t s_t^{\top}}{s_t s_t^{\top}} s_t \tag{5.2}$$

$$y_t^{\perp} = y_t - y_t^{=} \tag{5.3}$$

The aggregation step is performed applying a feed-forward network over the previous components, and obtaining the matrix $\mathbf{Y}^a$, where the rows are:

$$y_t^a = FF\left(\begin{bmatrix} y_t^{=} \\ y_t^{\perp} \end{bmatrix}; W_a\right) \tag{5.4}$$

Then a max-pooling operation over all words in $\mathbf{Y}^a$ is applied to obtain the `NIL` vector representation $\hat{n}$. Finally, a nil pointer score n_s is computed using a learnable weight $w_n^{\intercal}$, having $n_s = \hat{n}\mathbf{w}_n^{\intercal}$. The nil pointer score will be used to normalize the start and end pointer.

Answer Layer

NAMANDA uses a two stacked BiLSTMs on $\mathbf{Y}$ to determine the beginning and the end of the answer span. The hidden representations of these LSTMs are $\mathbf{B} \in \mathbb{R}^{T \times d}$, and $\mathbf{E} \in \mathbb{R}^{T \times d}$. The scores for the beginning s_b, and the end s_e, are then calculated using the question vector $\tilde{q}$, and the output of the stacked BiLSTMs $\mathbf{B}$, and $\mathbf{E}$.

$$s_b = \tilde{q}\,\mathbf{B}^{\intercal} \quad \text{and} \quad s_e = \tilde{q}\,\mathbf{E}^{\intercal} \tag{5.5}$$

Now, the `NIL` score is prepended, obtaining $\hat{s_b} = [n_s, s_b]$ and $\hat{s_e} = [n_s, s_e]$. The final spans are:

$$\begin{aligned} P(b|c,q) &= softmax(\hat{s_b}) \\ P(e|c,q) &= softmax(\hat{s_e}) \end{aligned} \tag{5.6}$$

And the answer joint probability is:

$$P(a|c,q) = P(b|c,q)P(e|c,q) \tag{5.7}$$

The `NIL` label is assigned when $b = 1$.

5.3 Word Representation

An important part of any NLP model is the word representation, we compare two solution (BERT and fastText) to find which of them is better suited for our multilingual task. We present how these two models create word representation and also how the multilingual versions are obtained.

5.3.1 BERT

Proposed by Devlin et al. (2019), BERT (Bidirectional Encoder Representations from Transformers) is a model for language representation. Similarly to (Radford et al., 2018), BERT uses a multi-layer transformer, with the addition that the transformer is bidirectional. The input representation is a sum over the token embedding, the position embedding, and the segment, which is the sentence encoding (the same for each token). BERT is pre-trained using as objective a Masked Language Model (MLM) and a next sentence prediction task. The MLM is based on the Cloze task (Taylor, 1953), the model is trained to predict the missing words that were randomly removed. This allows BERT to truly obtain a bidirectional representation. However, the use of a token to hide the word to predict creates discrepancies between the pretrain and the fine-tuning. Moreover, BERT assumes that the predicted token is independent from the other unmasked tokens, which ignores the high-order dependencies in natural languages (Yang et al., 2019).

The multilingual version has a shared embedding space for the top 100 languages with the largest Wikipedias. The entire Wikipedia dump for each language was taken as the training data for each language. The multilingual model has 12-layer, the internal hidden representation is of size 768, 12 attention heads, and a total of 110M parameters.

5.3.2 fastText

fastText (Bojanowski et al., 2017) learns n-grams representation and, using these representation, creates words representation as a sum of the n-grams is made of. fastText is based on the skip-gram with negative sampling model (Mikolov et al., 2013c), but with the use of subword information. In the negative sampling function (Equation 2.19), instead of using just the context word vector $\tilde{\mathbf{w}}_{t+j}$, we use the

subword information. Given a word x, $G_x \subset \{1, \ldots, G\}$ is the set of n-grams appearing in x. Each n-gram g has a corresponding vector representation $\mathbf{z}_g$. Thus, we obtain the SG model with negative sampling and subword information:

$$P\left(x_{t+j}|x_t\right) = \log \sigma \left(\sum_{g \in \mathcal{G}_{x_{t+j}}} \mathbf{z}_g^\top \mathbf{w}_t \right) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_z} \log \sigma \left(- \sum_{g \in \mathcal{G}_{x_{t+j}}} \mathbf{z}_g^\top \mathbf{w}_t \right) \quad (5.8)$$

Using the n-grams allows to better model morphological rich languages such as Finnish, and Turkish since even rare morphological inflections are modeled. Moreover, using n-grams allows to model unseen words since we can just sum the representation of the n-grams is made of.

The multilingual embeddings we use are obtained by training independent vector representations using fastText on each language's Wikipedia and common crawl corpora, then using a solution similarly to (Mikolov et al., 2013b) (Equation 2.27) the words are aligned into a common space.

Chapter 6

Experimental Evaluation

In this chapter we present the experiments that we performed to evaluate our system. We analyze the results, and present some problems of our approach.

Using our multilingual relation extraction dataset **X-WikiRE**, we evaluate two types of settings:

- **UnENT** in which the model is tested on unseen entities as in traditional relation extraction systems, and the zero-shot setting
- **UnREL** in which the model performance are evaluated on unseen relations (the entities may or may not been seen at training).

Our goal is to answer three different questions: **a)** how well relation extraction models can be transferred across different languages?, **b)** can the variance in the number of relations examples between languages help build more robust relation extraction models? and **c)** can a jointly trained model that performs RE on multiple languages perform equally or better than the monolingual trained version?

6.1 Experimental Setting

6.1.1 Metrics

We use the same evaluation methodology as Levy et al. (2017). We compute the precision, recall, and F1-score by comparing the spans of the answers predicted by the model with the ground truth answer span. The measure does not take in consideration the word order and punctuation but articles are not removed from the evaluation as separating them from entities is not as trivial for other languages as it is for English.

$$precision = \frac{\#\ of\ matching\ content\ words}{\#\ of\ content\ words\ in\ predicted\ answer} \tag{6.1}$$

$$recall = \frac{\#\ of\ matching\ content\ words}{\#\ of\ content\ words\ in\ ground\ truth\ answer} \tag{6.2}$$

$$F1\text{-}score = 2 * \frac{precison * recall}{precision + recall} \tag{6.3}$$

6.1.2 Hyperparameters

In all experiments, models were trained for five epochs with a learning rate of 1.0 using Adam (Kingma and Ba, 2015) as optimizer with $\beta_1 = 0.01, \beta_2 = 0.999$, $\epsilon = 1e - 8$. For finetuning in the cross-lingual transfer experiments, the learning rate was lowered to 0.001 to prevent forgetting and a maximum of 30 iterations over the small target language training set were performed with model selection using the target language development set F1-score.

The embeddings used are from the multilingual fastText, except when comparing against BERT and when confronting on the dataset of Levy et al. (2017), where we use GloVe to maintain the compatibility of results.

6.2 Word Representation

We compared the two multilingual word embedding models presented in Section 5.3 to find which of them is the best solution for our task. To compare these two solution we used the *Cross-lingual Model Transfer* setup described in Section 5.1.1. The model is first trained using 1M examples on a high resource language (English) then fine-tuning on the target language is performed. The fine-tuning is performed with different amount of examples, the settings are: zero-shot (0 examples), 1K, 2K, 5K, and 10K. Table 6.1 shows the results for our model in the **UnENT** scenario using both multilingual BERT and fastText. As we can see, BERT performs poorly compared to fastText in every language and almost for each of the fine-tuning settings.

A reason of this can be the high difference in word coverage. As shown in Table 6.2, the vocabulary coverage for the contexts and questions is double for all languages for fastText compared to BERT. This is caused by the disparity in the number of words in the vocabulary of the embeddings. BERT has only 120K tokens for 104 languages, while fastText's vocabulary contains around 100K tokens for each language.

	Setting	BERT			fastText		
		P	R	F1	P	R	F1
EN-IT	0	9.76	11.84	**10.70**	54.86	2.31	4.44
	1K	64.15	50.08	56.25	69.25	59.53	**64.02**
	2K	70.89	57.12	63.26	80.78	68.89	**74.37**
	5K	79.85	68.33	73.64	83.77	81.90	**82.82**
	10K	83.39	76.97	80.06	84.43	82.50	**83.45**
EN-ES	0	10.22	3.75	5.49	28.98	11.22	**16.17**
	1K	55.19	35.19	42.97	67.56	64.08	**65.78**
	2K	68.43	50.36	58.02	74.45	72.73	**73.58**
	5K	72.65	63.97	68.04	79.23	74.03	**76.54**
	10K	79.76	65.06	71.66	79.27	76.76	**77.99**
EN-FR	0	23.72	13.77	**17.42**	49.56	9.03	15.28
	1K	53.81	35.38	42.69	69.88	61.93	**65.67**
	2K	67.51	52.74	59.21	77.30	63.37	**69.64**
	5K	70.98	66.16	68.49	76.35	69.11	**72.55**
	10K	78.67	67.11	72.43	80.78	68.52	**74.15**
EN-DE	0	3.33	2.52	2.87	21.00	10.60	**14.09**
	1K	50.84	62.35	56.01	55.84	70.91	**62.47**
	2K	58.46	65.96	61.99	56.67	75.13	**64.60**
	5K	61.37	71.75	66.16	68.00	76.24	**71.88**
	10K	66.65	74.66	70.43	66.89	78.37	**72.17**

Table 6.1: Precision, Recall and F1-scores for **UnENT** comparing scores using BERT and fastText multilingual embeddings.

	BERT		fastText	
Lang.	Context	Question	Context	Question
IT	24%	35%	64%	71%
FR	25%	36%	67%	73%
ES	24%	37%	65%	73%
DE	22%	34%	56%	64%
EN	30%	37%	63%	71%

Table 6.2: Vocabulary coverage for multilingual BERT and fastText.

6.3 Transfer Setups

We perform experiments of the two previously described architectures to see how well each scenario perform in different types of multilingual and cross-lingual transfer. We also present the results for the monolingual setup, and use it as a comparison for the multilingual models.

6.3.1 Monolingual

We train a model for each language on the full monolingual training set (1 million of instances), the model will be used as a comparison for the multilingual and the cross-lingual transfer models. Table 6.3 presents the results obtained for both the **UnENT** and **UnREL** scenarios. The monolingual should be seen as a ceiling, as the other models will be trained with less data.

Lang.		UnENT	UnREL
EN	P	74.09	46.75
	R	85.35	25.32
	F1	79.32	32.78
ES	P	81.79	49.77
	R	85.02	27.69
	F1	83.37	34.54
IT	P	88.69	47.09
	R	88.10	29.45
	F1	88.39	35.62
FR	P	82.36	42.93
	R	74.16	25.73
	F1	78.05	31.78
DE	P	75.85	41.94
	R	88.21	24.38
	F1	81.57	30.82

Table 6.3: Precision, Recall, and F1-scores for the monolingual baseline for all languages in the **UnENT** and **UnREL** setup.

6.3.2 Cross-lingual Model Transfer

In this set of experiments, we test how well RE models can be transferred from a source language with a large number of training examples to target languages with no or minimal training data (question **a**).

In the **UnENT** experiments, we construct pairwise parallel test and development sets between English and each of the languages, meaning that the entities that are in the English and the target train/dev set are non present in the target test set. Then an English relation extraction model, built on top of the multilingual representations, is trained on a full English training set (1 million instances). A similar approach is followed also for the **UnREL** scenario, in this case the train/dev sets do not contain any relation that is in the target's language test set.

The results presented in Table 6.2 are for the cross-lingual transfer **UnENT** experiments. While it is clear that the models suffer from rather low recall when no fine-tuning is performed, the results show considerable improvements when

performing fine-tuning with just 1000 target language examples. When using 10K target language examples, it is possible to nearly match the performance of a model trained on the full target language monolingual training set. In Figure 6.1, we can have better view of the performances when using fastText.

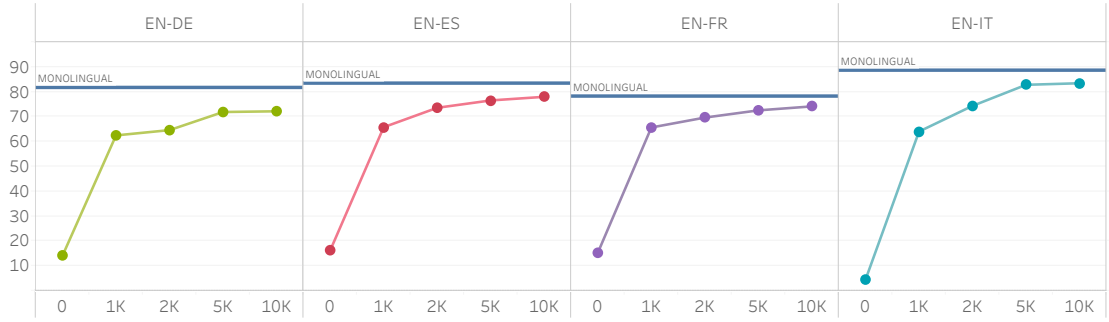


Figure 6.1: F1-scores for the cross-lingual transfer experiments in the **UnENT** setting. The MONOLINGUAL line shows the corresponding monolingual model's F1-score.

The **UnREL** is a considerably more challenging setting, we are interested in evaluating along a different dimension (question **b**): when relations are seen in the source language but not in the target language. Figure 6.2 shows the results for the **UnREL** setup. In this case, we perform the transfer with 10K examples, and similarly to **UnENT**, the results show that it's possible to recover a large part of the fully-supervised monolingual models' performance, but not as much as before. This indicates that it is more challenging to transfer the ability to identify relation paraphrases and entity types through global indications (when context phrasing differs from the question in a way that is shared between relations) which Levy et al. (2017) suggested are important for generalizing to new relations in this framework.

6.3.3 One Model, Multiple Languages

We now evaluate the model jointly trained on multiple languages. We are interested in exploiting the fact that KBs are better populated for different properties across different languages (question **c**) . We perform a 5-folds cross-validation, in each experiment a test set relation for a particular language is not seen in that language's training set, but may be seen in any of the other languages. This amounts to maintaining the original zero-shot setting (where a relation is not seen) monolingually, but providing supervision by allowing the models to *peek across languages*. To control for training set size we include 200K training instances per language

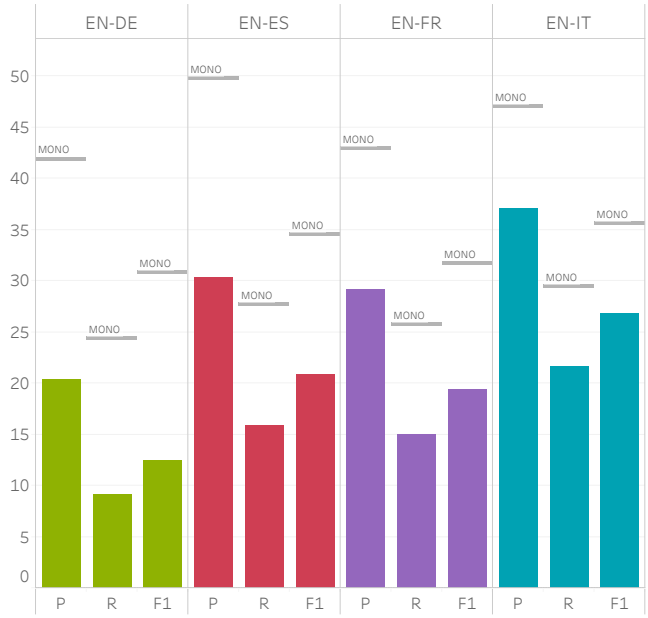


Figure 6.2: Precision, Recall and F1-scores for the cross-lingual transfer experiments in **UnREL** setting. The results are the mean of 5-fold cross-validation. The MONO line shows the corresponding monolingual model's F1-score.

(Multi. (S)), so that the total size of the training set is equal to that of the monolingual baseline. However, an additional benefit of multilingual training is that extra overall training data becomes available. To test the effect of that we also run an experiment where the full training set of each of the languages is employed, adding up to 5 million training examples (Multi. (L)).

In the **UnENT** setting the multilingual models trained on just 200k instances per language perform slightly below the monolingual baselines. This excludes for French where, surprisingly, the baseline performance is actually exceeded. When the full training sets of all languages are combined, the multilingual model outperforms the monolingual baselines for English, Spanish, and French and is slightly worse for German and Italian. This demonstrates that not only is it possible to utilize a single model to perform RE in multiple languages, but that the multilingual supervision signal will often lead to improvements in performance. These results are shown in the third and fourth columns of Table 6.4.

The multilingual **UnREL** model outperforms its monolingual counterparts by large margins for all languages. For many of the languages the improvements in F1-score are doubled. This is largely in line with our premise that the natural topicality of KBs across languages can be exploited to provide cross-lingual supervision for relation extraction models.

Lang.		UnENT			UnREL	
		Mono.	Multi. (S)	Multi. (L)	Mono.	Multi.
EN	P	74.09	74.33	**77.11**	46.75	**63.29**
	R	85.35	83.63	**86.42**	25.32	**44.40**
	F1	79.32	78.71	**81.50**	32.78	**51.99**
ES	P	81.79	80.60	**83.68**	49.77	**73.43**
	R	**85.02**	81.47	83.58	27.69	**62.82**
	F1	83.37	81.03	**83.63**	34.54	**67.64**
IT	P	**88.69**	86.23	88.43	47.09	**68.66**
	R	**88.10**	85.64	86.91	29.45	**55.24**
	F1	**88.39**	85.93	87.66	35.62	**61.13**
FR	P	82.36	80.82	**82.90**	42.93	**60.78**
	R	74.16	76.60	**78.10**	25.73	**47.09**
	F1	78.05	78.66	**80.43**	31.78	**53.06**
DE	P	**75.85**	69.88	73.67	41.94	**43.36**
	R	**88.21**	81.36	84.08	24.38	**25.32**
	F1	**81.57**	75.20	78.53	30.82	**31.97**

Table 6.4: Precision, Recall, and F1-score results for all languages' monolingual (Mono.) and multilingual (Multi.) models. (S) indicates the small multilingual model which was trained on 200K examples and (L) indicates the large on trained on 5 million examples.

6.4 Reading Comprehension Model

To evaluate how the reading comprehension model affects the performances, we compared NAMANDA (Monolingual) with the nil-aware BiDAF (Levy et al., 2017). For the comparison we used their dataset, and trained both system in a monolingual fashion using GloVe embeddings. The results presented in Table 6.5 are in line with these presented in (Kundu and Ng, 2018a), however these results are higher than when using **X-WikiRE**, this is a consequence of the differences in the datasets, the embeddings, and the training. **X-WikiRE** context are a bit longer than these in Levy et al. (2017) (Table 4.3), moreover the coverage of GloVe is higher. Also, the main difference is that in our setup we only used 5-folds instead of 10-folds, meaning that the model sees less unique relations during the training phase and is also tested on more.

Lang.		UnENT		UnREL	
		Levy et al. (2017)	Monolingual	Levy et al. (2017)	Monolingual
EN*	P	87.66	**90.49**	43.61	**56.53**
	R	91.32	**94.87**	36.45	**44.74**
	F1	89.44	**92.63**	39.61	**49.85**

Table 6.5: Comparison with Levy et al. (2017). The models are compared on their monolingual dataset in English (EN*).

Chapter 7

Conclusions

Given the widely lamented fact that KBs and resources are highly skewed towards English causing difficulties in the development of NLP system for low-resource languages, in this thesis we introduced a new large-scale multilingual dataset (**X-WikiRE**) for relation extraction. It is a reading comprehension-based relation extraction dataset for five languages: English, German, French, Spanish, and Italian. We hope that **X-WikiRE** will facilitate the research on multilingual methods for relation extraction. **X-WikiRE** is at least an order of magnitude larger than previous multilingual datasets. We used **X-WikiRE** to experiment with various transfer learning setup using a state-of-the-art nil-aware machine reading comprehension model to perform zero-shot relation extraction. We evaluated the use of our resource using two different setups: transfer learning and joint learning. The experiments demonstrated that:

a) multilingual training can be employed to exploit the fact that KBs are better populated in different areas for different languages, providing a strong cross-lingual supervision signal which leads to considerably better zero-shot relation extraction;
b) models can be transferred cross-lingually with a minimal amount of target language data for fine-tuning;
c) better modelling of nil-awareness in reading comprehension models leads to improvements on the task.

Our work is a step towards making KBs equally well-resourced across languages. Future work may include the use of better multilingual embeddings with higher coverage, as we saw affects the performances in the zero-shot scenario drastically. Moreover, it will be interesting to expand the dataset to more languages, in particular to true low-resourced ones.

Bibliography

Mostafa Abdou, Artur Kulmizev, Vinit Ravishankar, Lasha Abzianidze, and Johan Bos. 2018. What can we learn from semantic tagging? In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4881–4889. Association for Computational Linguistics.

Zeynep Akata, Florent Perronnin, Zaïd Harchaoui, and Cordelia Schmid. 2016. Label-embedding for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1425–1438.

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *CoRR*, abs/1812.10464.

Akari Asai, Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2018. Multilingual extractive reading comprehension by runtime machine translation. *CoRR*, abs/1809.03275.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Nguyen Bach and Sameer Badaskar. 2007. A Review of Relation Extraction. In *Literature review for Language and Statistics II, 2*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2670–2676.

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 28–36. The Association for Computer Linguistics.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 932–938. MIT Press.

Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Marcel Bollmann, Natalia Korchagina, and Anders Søgaard. 2019. Few-shot and zero-shot learning for historical text normalization. *CoRR*, abs/1903.04870.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Machine Learning, Proceedings of the Tenth International Conference, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*, pages 41–48. Morgan Kaufmann.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 160–167. ACM.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating Cross-lingual Sentence Representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2018. Universal transformers. *CoRR*, abs/1807.03819.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie M. Strassel. 2015. Overview of linguistic resources for the TAC KBP 2015 evaluations: Methodologies and results. In *Proceedings of the 2015 Text Analysis Conference, TAC 2015, Gaithersburg, Maryland, USA, November 16-17, 2015, 2015*. NIST.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Manaal Faruqui and Shankar Kumar. 2015. Multilingual Open Relation Extraction Using Cross-lingual Projection. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1351–1356. Association for Computational Linguistics.

Orpaz Goldstein. 2018. Zero-shot relation extraction from word embeddings. Master's thesis, University of California, Los Angeles, CA, USA.

Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2017. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232.

Michael Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. WikiReading: A Novel Large-scale Language Understanding Task over Wikipedia. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1545. Association for Computational Linguistics.

Sepp Hochreiter. 1991. *Untersuchungen zu dynamischen neuronalen Netzen*. Ph.D. thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Portland, Oregon, USA. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Michael I. Jordan. 1986. Serial order: A parallel distributed processing approach. In John W. Donahoe and Vivian Packard Dorsel, editors, *Neural-Network Models of Cognition*, volume 121 of *Advances in Psychology*, pages 471 – 495. North-Holland.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 908–918, Berlin, Germany. Association for Computational Linguistics.

Lucie-Aimée Kaffee and Elena Simperl. 2018. Analysis of editors' languages in wikidata. In *Proceedings of the 14th International Symposium on Open Collaboration, OpenSym 2018, Paris, France, August 22-24, 2018*, pages 21:1–21:5.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain, July 21-26, 2004 - Poster and Demonstration*. ACL.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Souvik Kundu and Hwee Tou Ng. 2018a. A nil-aware answer extraction framework for question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4243–4252, Brussels, Belgium. Association for Computational Linguistics.

Souvik Kundu and Hwee Tou Ng. 2018b. A question-focused multi-factor attention network for question answering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5828–5835. AAAI Press.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. 2008. Zero-data learning of new tasks. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 646–651. AAAI Press.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-Shot Relation Extraction via Reading Comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342. Association for Computational Linguistics.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Neural relation extraction with multi-lingual attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34–43, Vancouver, Canada. Association for Computational Linguistics.

Angli Liu, Stephen Soderland, Jonathan Bragg, Christopher H. Lin, Xiao Ling, and Daniel S. Weld. 2016. Effective crowd annotation for relation extraction. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 897–906, San Diego, California. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048. ISCA.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Erik Gundersen Miller. 2002. *Learning from one example in machine vision by sharing probability densities*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*. Society for Artificial Intelligence and Statistics.

Abiola Obamuyide and Andreas Vlachos. 2018. Zero-shot relation classification as textual entailment. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 72–78, Brussels, Belgium. Association for Computational Linguistics.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Sachin Pawar, Pushpak Bhattacharyya, and Girish Palshikar. 2017. End-to-end relation extraction using neural networks and markov logic networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 818–827, Valencia, Spain. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1498–1507, Sofia, Bulgaria. Association for Computational Linguistics.

Hoifung Poon, Janara Christensen, Pedro Domingos, Oren Etzioni, Raphael Hoffmann, Chloe Kiddon, Thomas Lin, Xiao Ling, Mausam, Alan Ritter, Stefan

Schoenmackers, Stephen Soderland, Dan Weld, Fei Wu, and Congle Zhang. 2010. Machine reading at the university of Washington. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 87–95, Los Angeles, California. Association for Computational Linguistics.

Boyu Qiu, Xu Chen, Jungang Xu, and Yingfei Sun. 2019. A survey on neural machine reading comprehension. *CoRR*, abs/1906.03824.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia. Association for Computational Linguistics.

Ellen Riloff and Michael Thelen. 2000. A rule-based question answering system for reading comprehension tests. In *Proceedings of the 2000 ANLP/NAACL Workshop on Reading Comprehension Tests As Evaluation for Computer-based Language Understanding Sytems - Volume 6*, ANLP/NAACL-ReadingComp '00, pages 13–19, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129, Denver, Colorado. Association for Computational Linguistics.

Sebastian Ruder. 2019. *Neural Transfer Learning for Natural Language Processing*. Ph.D. thesis, National University of Ireland, Galway.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Latent Multi-task Architecture Learning. *arXiv e-prints*, page arXiv:1705.08142.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

M. Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual NLP. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1713–1722, Beijing, China. Association for Computational Linguistics.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Wilson L. Taylor. 1953. "Cloze Procedure": A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the*

48th Annual Meeting of the Association for Computational Linguistics, pages 384–394, Uppsala, Sweden. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual Relation Extraction using Compositional Universal Schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 886–896. Association for Computational Linguistics.

Patrick Verga and Andrew McCallum. 2016. Row-less universal schema. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 63–68, San Diego, CA. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1340–1349, Osaka, Japan. The COLING 2016 Organizing Committee.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. 2017. Zero-shot learning - the good, the bad and the ugly. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3077–3086. IEEE Computer Society.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, pages 1006–1011, Denver, Colorado. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, and Ruslan Salakhutdinov. 2017. Words or characters? fine-grained gating for reading comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1456–1466, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 419–426, Ann Arbor, Michigan. Association for Computational Linguistics.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 427–434, Ann Arbor, Michigan. Association for Computational Linguistics.