Karlsruhe Institute of Technology

Inspection of Multilingual Neural Machine Translation

Bachelor's Thesis
of

Carlos Mullov

At the Department of Informatics
Institute for Anthropomatics and Robotics (IAR)

Primary referee:	**Prof. Dr. Alexander Waibel**
Secondary referee:	**Prof. Dr. Tamim Asfour**
Advisor:	**Dr. Jan Niehues**

Duration: September 7th, 2017 – December 6th, 2017

Erklärung:

Ich versichere hiermit, dass ich die Arbeit selbstständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis beachtet habe.

Karlsruhe, den 6. Dezember 2017

Carlos Mullov

Abstract:

In this thesis we explore the ability of the multilingual attention-based encoder-decoder NMT system proposed by Ha et al. (2016) to produce a language independent representation for the sentence it translates. NMT systems are trained to produce translations using large amounts of training data in the form of bilingual texts, the so called parallel corpora. However, for most language pairs high quality training resources are often too scarce in order to sufficiently train an NMT system. The ability to abstract from individual languages by using a shared representation across all languages would alleviate the need for large parallel corpora for every individual language pair. Having put forward the hypothesis, that with better abstraction ability of the NMT system the hidden representation of the sentence becomes more independent of the language pair that was involved in its generation, we have explored this independence by measuring the degree of correlation between the hidden representation and the language pair. Of the various hidden representations in the translation system we have chosen to examine the so called *context vectors*. In order to measure this correlation, we have built classifiers based on feed-forward neural networks which, given a context vector, attempt to predict the language pair involved in its generation. Having trained multiple multilingual NMT systems with 5 languages on the source side as well as the target side, we have trained single layer perceptrons to classify the context vectors produced by these systems. Those classifiers have achieved rates of correct classification of up to 95.75% for 25 possible classes, which indicates that our NMT system does not successfully abstract from the languages involved in the translation. Furthermore, having explored the underlying cause of success in classification, we have found present a linear relation in Euclidean space between context vectors of different languages. More precisely, for a pair of languages (A,B) we have found a vector b_{AB}, such that for a context vector c with A as its source language, $c+b_{AB}$ is classified as having B as the source language in 90.9% of the cases. We have found this translation for the source language to be independent of the target language.

In conclusion, despite the sentence representation in our NMT systems not being independent of the individual languages, for this representation we have found it to be surprisingly easy to transform the features representing the source language, which potentially could be made use of in order to improve zero-shot translation.

Kurzzusammenfassung:

In dieser Arbeit untersuchen wir die Fähigkeit des von Ha et al. (2016) beschriebenen multilingualen attention-basierten encoder-decoder NMT-Systems, während der Übersetztung eine sprachenunabhängige Repräsentation des übersetzten Satzes zu erzeugen. NMT-Systeme werden mithilfe von großen Mengen an Trainingsdaten in der Form von bilingualen Texten, sogenannten Parallelcorpora, trainiert Übersetzungen zu erzeugen. Für die meisten Sprachpaare sind hochqualitative Trainingsressourcen jedoch zu knapp, um mit ihnen ein NMT-System zu trainieren. Die Fähigkeit mithilfe von einer gemeinsamen Repräsentation von individuellen Sprachpaaren zu abstrahieren, würde die Notwendigkeit für große Parallelcorpora für jedes einzelne Sprachpaar beseitigen. Mit der Hypothese, dass mit zunehmender Abstraktionsfähigkeit des NMT-Systems die Unabhängigkeit der Satzrepräsentation von dem Sprachpaar, welches in ihrer Erzeugung involviert war steigt, haben wir das diese Unabhängigkeit untersucht, indem wir das Maß der Korrelation zwischen der Repräsentation und dem Sprachpaar gemessen haben. Von den verschiedenen intermediären Representation in dem NMT-System haben wir uns dazu entschieden die *Kontextvektoren* zu untersuchen. Dieses Maß an Korrelation haben wir mithilfe von feed-forward Neuronalen Netzen gemessen, welche, gegeben ein Kontextvektor, versuchen das Sprachpaar aus dem er erzeugt wurde zu bestimmen. Für die Evaluation haben wir mehrere multilinguale NMT-Systeme und einschichtige multilayer-Perceptronen, die die von den NMT-Systemen erzeugenen Kontextvektoren klassifizieren trainiert. Diese Klassifikatoren haben bei einer Gesamtzahl von 25 Klassen Klassifikationsraten von bis zu 95.75% erzielt, was darauf hinweist, dass unsere NMT-Systeme nicht von den individuellen Sprachen abstrahiert. Bei der Untersuchung des Grundes für die erfolgreiche Klassifikation, haben wir eine lineare Beziehung zwischen Kontextvektoren verschiedener Sprachen im euklidischen Raum gefunden. Genauer, haben wir für beliebiges Sprachpaar (A,B) einen Vektor b_{AB} gefunden, sodass für den Kontextvektor c mit A als Quellsprache $c+b_{AB}$ in 90,9% der Fälle mit B als Quellsprache klassifiziert wurde. Weiterhin ist diese Verschiebung unabhängig von der Zielsprache und beeinflusst diese nicht.

Schlussendlich haben wir trotz Abhängigkeit der Satzrepräsentation von den einzelnen Sprachen feststellen können, dass für diese Repräsentation die Merkmale, die die Quellsprache darstellen überraschend einfach zwischen verschiedenen Quellsprachen abänderbar ist, was möglicherweise für die Verbesserung von zero-shot-Übersetzung verwedet werden könnte.

Contents

1. Introduction

1.1. Introduction

Since the introduction of neural machine translation (NMT) in recent years, the field of machine translation has made significant progress. Since then various architectures for NMT systems have been proposed, aiming to improve translation quality and further improving the effectiveness, inter alia by adding multilinguality.

Many of the current state-of-the-art NMT systems are based on the attention-based encoder-decoder architecture proposed by Bahdanau et al. (2014). As the name implies the NMT system consists of an encoder, a decoder and an attention mechanism "in between them". The encoder encodes the input sentence to be translated into a neural representation – a set of fixed-size vectors. With the help of the attention mechanism the decoder word by word produces the translated sentence from this representation: for each decoder step the attention mechanism produces a word alignment for the word to be generated towards each word in the source sentence, and creates another representation based on this alignment. The attention mechanism then provides this representation to the decoder in order for the decoder to generate the word. Both encoder and decoder consist of recurrent neural networks, hence the name neural machine translation. This set of vectors is known as *context vectors*.

At the time of proposition, the system was designed for bilingual translation only. However, with this system as basis Ha et al. (2016) have proposed an approach to extend this architecture to multilingual many-to-many translation. This is achieved by training the system without any modifications to the architecture, with unified vocabularies and training corpora from multiple languages, resulting in a single translation system which learns to translate from multiple languages to multiple languages. The expectation with this approach is that the system learns common semantics across multiple languages by finding a common neural representation for sentences of different languages. This ability of abstract from individual languages would further the NMT system generalization ability and ultimately result in an in improvement of translation quality, while reducing the amount of training data needed for each individual language pair. Since most of the time parallel training data for pairs of "exotic" languages is extremely scarce, this would help improve availability of machine translation for situations where translation between such languages is needed.

This concept of encoding sentences of multiple different languages into one shared representation and then decoding it into a variety of target languages closely resembles the concept of interlingua-based translation: an interlingua, a concept in machine translation, is a language without any ambiguity and with the capacity to store all of the extracted meaning from a sentence of any other language. This allows for translation of a sentence from any language into a sentence in the interlingua without the loss of information and then for translation of that interlingua representation into a sentence of another language. Such an approach is also comparable to the frequently used concept of a pivot language, according to which multilingual translation is achieved by first translating a source sentence into a pivot language, such as English, and then translating the resulting English sentence into the desired target language. However, contrary to the interlingua-based translation this generated new mistakes in translation, due to ambiguities in the pivot language.

In the proposed multilingual NMT system, the better the system learns to extract common features of different languages, the closer the hidden representation of different languages becomes, ultimately resulting in a zero-distance between the representations of two sentences with different languages but with the same meaning. In other words, the resulting hidden representation of a sentence would, ideally, perfectly abstract from the individual source and target languages, arriving at the same representation of sentences of the same meaning and thus from this representation the decoder would thus generate the

exact same target sentences. The vector space induced by the context vectors in such a translation system would thus be an interlingua.

In this thesis we shall examine the resulting hidden representation in a translation system as proposed by Ha et al. (2016) in order to test how well the NMT system learns the abstraction from languages. For this hidden representation we choose the *context vectors*. As with increasing abstraction ability of the NMT system the context vectors become more independent of the languages that were involved in their generation, we believe this independence to be an indicator worth investigating. With the context vectors increasing independence the correlation between the languages and the context vectors would decrease, with no discernible correlation in the scenario described above. Therefore we shall test for the independence by trying to correlate a given context vector with the language pair that was involved in its generation. This correlation we will produce through neural network-based classification of the context vectors.

2. Background

2.1. Artificial Neural Networks

Artificial neural networks (ANN) are a machine learning model which has recently become quite popular as it has been delivering good results. Consequently, ANNs are now applied in many fields, one of them being natural language processing (NLP), as they have been often found to show improved results over previously used techniques. Machine translation as part of this field of study, has demonstrated improved results compared to statistical machine translation (SMT), which has been the state-of-the-art technique prior to the introduction of neural machine translation (NMT).
ANNs are said to have been inspired by biological neural networks and are commonly regarded as powerful function estimators. There are numerous variations of ANNs, and for this thesis we will be taking a look at *feed forward neural networks* and *recurrent neural networks*.

2.1.1. Feed Forward Neural Networks

Feed forward neural networks (FFNN) were the first type of ANN, originating from the perceptron, the simplest type of neural network. Taking a vector $x \in \mathbb{R}^n$ with fixed dimension n as input, they have been proven to be able to approximate any arbitrary function $f : \mathbb{R}^n \to \mathbb{R}^m$. FFNNs are commonly represented as weighted directed acyclic graphs. Its nodes, called neurons are partitioned into layers, which have a fixed ascending order. Those layers are categorized into three types:

Input layer the first layer. Contains n neurons. Each neuron assumes the value of the corresponding entry of the input vector.

Output layer the last layer. Contains m neurons. The values assumed by the neurons in this layer after a forward pass represent the network output.

Hidden layer every remaining layer between the input and the output layers.

FFNNs have one input layer, one output layer, an arbitrary amount of hidden layers between them, and weighted connections between the neurons of the adjacent layers. Given an input, the network then calculates an output by successively performing a *forward pass* for each layer: starting with the layer after the input layer, the output o_i for each layer l_i is calculated by taking the previous layer output o_{i-1} as the input for the current layer, and subsequently performing the forward pass. After such an operation for each layer, the resulting vector in the output layer is then the FFNN output.

Classical FFNNs, often called *multilayer perceptron* (MLP), have fully connected layers, meaning that each neuron of a layer is connected to each and every neuron of the adjacent layers. With n_{k-1} neurons in the $(k-1)$-th layer and n_k neurons in the k-th layer we get $n_{k-1} \cdot n_k$ weighted connections between those layers, which are represented by a weight matrix $W \in \mathbb{R}^{n_{k-1} \times n_k}$.
The i-th neuron output in the k-th layer for this kind of neural network is calculated as

$$o_i = \sigma\left(\sum_{j=1}^{n_{k-1}} w_{ij} x_j\right) \tag{2.1}$$

where x is the $(k-1)$-th layer output and σ is the layer activation function.
The activation function is a nonlinear and differentiable function, such as the commonly used *sigmoid*-function or *tanh*-function.

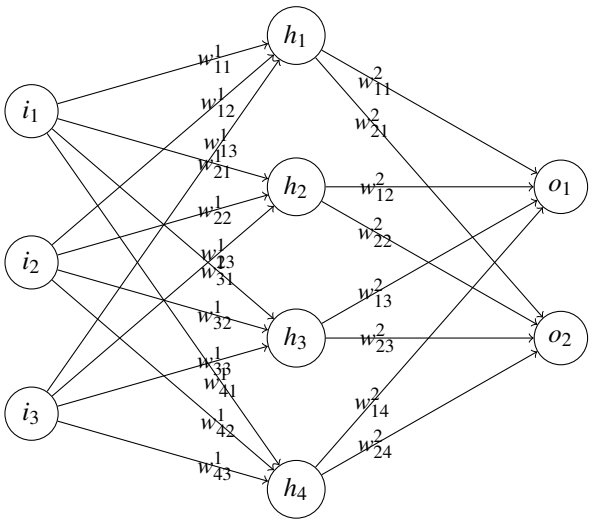

Figure 2.1.: A Simple Feed Forward Network with one hidden layer, 3-dimensional input, 2-dimensional output and fully connected layers

In this thesis we will call the special case of a MLP with no hidden layers *single layer perceptron* (SLP). Aside from MLPs there are further types of FFNNs, such as *time delayed neural networks* (TDNN, Waibel et al., 1989) and *convolutional neural networks*, which aside from fully connected layers also use different types of layers.

2.1.2. Supervised Neural Network Training

In this thesis neural networks will be used in supervised learning exclusively. In supervised learning the neural network is trained using labeled data. Such labeled data consists of pairs of training examples and corresponding labels, indicating the expected network output. For classification this label will usually indicate the correct class of a training example and for translation it will usually be a human translation of the of the example sentence (see section 2.2.5). Neural networks are trained by optimizing the network parameters (weights and biases), such that the network produces the desired output.

This is most commonly achieved by the use of a gradient descent technique called backpropagation. By choosing an appropriate loss-function E, which, given the network output and the label, calculates the error, we can calculate the contribution of each weight to this error, via

$$\frac{\partial E}{\partial w_{ij}}$$

Choosing a learning rate η, the network weights are then iteratively optimized, via

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} \tag{2.2}$$

for each value-label pair in the training set. In this method the parameter space is systematically searched for parameters which minimize the loss-function, converging towards a local minimum. The learning rate η affects the step size for a weight update, leaning towards faster convergence for high values and towards more accurate training for low values.

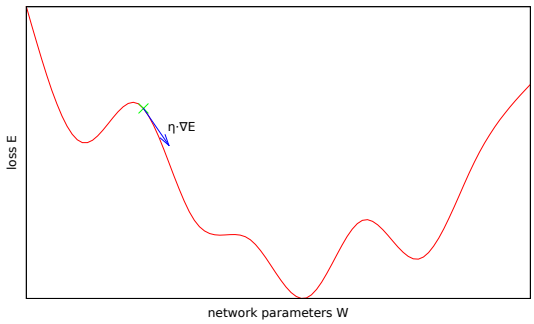

Figure 2.2.: Gradient descent iteratively optimizes the network parameters, converging to a local minimum of the loss function

Commonly used loss-functions include the *summed squared error*

$$sse(x,y) = \frac{1}{2}\sum_{i=0}^{n}(x_i - y_i)^2 \tag{2.3}$$

which is often used for regression and the *cross entropy error*

$$H(x,y) = -\sum_{i=0}^{n} y_i \log(x_i) \tag{2.4}$$

which measures the loss for probability distributions. In training x is the network output, and y is the label for the input which produced x.

2.1.3. Classification

Classification is the task of determining the correct class for a given data point in a feature space, partitioned into classes and is a very common task for ANNs. A common approach for classification with neural networks, is to interpret the network output as probability distribution, with each dimension indicating the probability of the input belonging to the corresponding class. This can be achieved by choosing the right output activation function and loss function, such as softmax in combination with cross entropy error.

Linear Separability

A set of data points D partitioned into two classes $\{C_0, C_1\}$ is linearly separable if there exists a hyperplane, defined via $wx + b = 0$, such that

$$\forall x \in C_0 : wx + b < 0$$
$$\forall x \in C_1 : wx + b \geq 0$$

In case of the classification with n classes this definition can be extended to a separation of the data points with n separate hyperplanes. This definition is equivalent to the existence of an MLP with no hidden layers, which correctly classifies the data points, as such an MLP defines a hyperplane for each of the output neurons (in case of the use of a bias neuron in the input layer). Thus successfully training such a network to produce significantly high classification accuracies, implies the linear separability of a given data set.

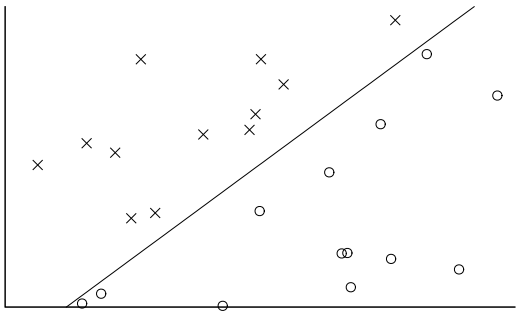

Figure 2.3.: Example of a linearly separable set of data points with two classes represented by circles and crosses

2.1.4. Recurrent Neural Networks

Recurrent Neural Networks (RNN) are a variant of ANNs, with variable length input and the ability to remember the input context. As Sutskever et al. (2014) describe it, "The Recurrent Neural Network is a natural generalization of feedforward neural networks to sequences". RNNs calculate their output in multiple steps, while remembering the context via a state. For an input sequence $x = (x_1, \ldots, x_n)$, in each step t the network f calculates the new state

$$h_t = f(x_t, h_{t-1}) \tag{2.5}$$

Classical RNNs typically consist of one input layer, one hidden layer and one output layer and the state for each step is the hidden layer value. The hidden layer is fully connected to the concatenation of the current step input and the previous state. Thus, in matrix notation, equation 2.5 for such RNNs resolves to

$$\vec{h_i} = \sigma(W\vec{x_i} + U\vec{h}_{i-1}) \tag{2.6}$$

$\vec{x_i}$ being the i-th vector from the input sequence, W being the weights between the input and the hidden layer, U being the weights between the previous state and the hidden layer, and σ being the hidden layer activation function. The initial hidden state h_0 is usually the null vector.

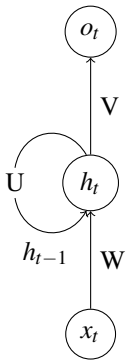

Figure 2.4.: A Recurrent Neural Network passes its previous hidden state when calculating the next hidden state, thus remembering the input context

RNNs have been employed to many fields, including natural language processing, showing good results. However, classical RNNs are currently rarely used, as they have difficulties remembering long

term dependencies. Instead, more recent variants of RNNs are commonly applied, such as the *long-short term memory* (LSTM, Hochreiter and Schmidhuber, 1997) and *gated recurrent unit* (GRU, Cho et al., 2014a), which both use a mechanism called *gates* to better learn long term dependencies.

2.1.5. Backpropagation Through Time

RNNs also are trained by means of a variation of backpropagation, called Backpropagation Through Time (BPTT). In order to apply standard backpropagation to RNNs, after its application to an input the RNN is then converted into a network equivalent to a FFNN, by means of a technique known as "unfolding". The recurrence is resolved by building layerwise an FFNN by using the history of RNN states in each timestep. After the unfolding, the same weights are found present in multiple different layers of the resulting FFNN. As a result in the process of backpropagation multiple different weight updates are assigned to the same weight. To resolve this, the principle of *shared weights* is applied: the actual weight update is calculated from the mean of the different weight updates across the unfolded weights.

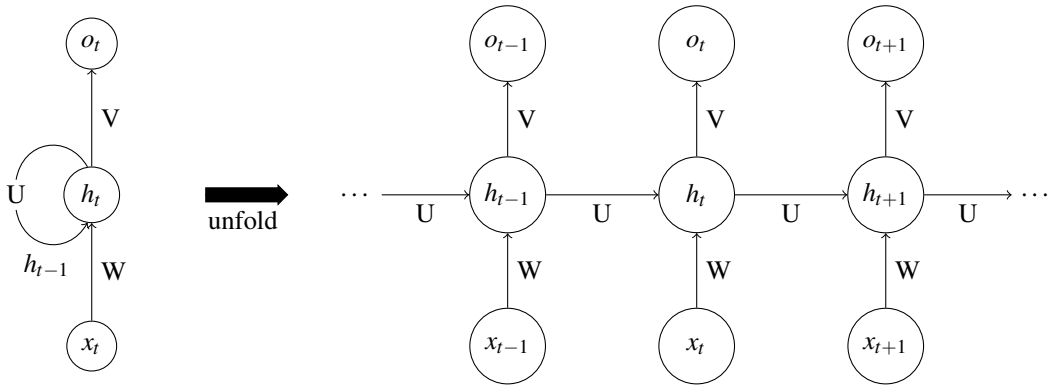

Figure 2.5.: In backpropagation through time in order to apply backpropagation to an RNN, the RNN is unfolded after having been applied to an input sequence. As a result the originally unique weights are found present multiple times.

2.2. Neural Machine Translation

Machine translation is the task of finding the most appropriate translation in the target language, for a sentence in the source language. As illustrated by the Vauquois Triangle (see figure 2.6), there are several linguistic approaches to this problem, the most direct being the direct mapping of the words in the source sentence to their respective counterparts in the target language. With increasing depth of analysis of the source sentence the translation process abstracts more from the language pairs involved, producing a language independent representation in the translation process. The *interlingua translation* represents the translation approach with the highest depth of analysis.

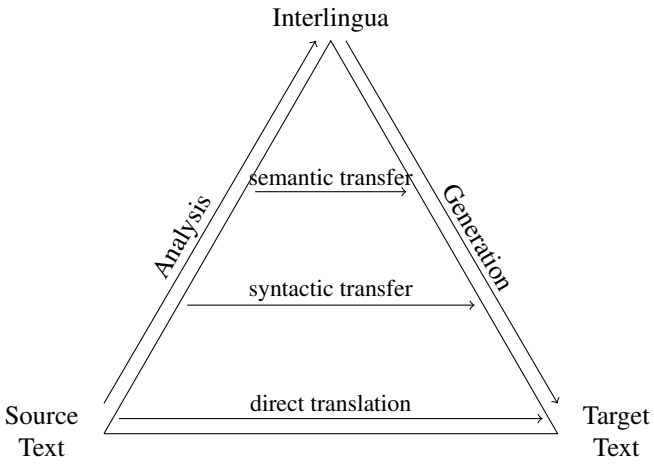

Figure 2.6.: The Vauquois Triangle illustrates the different linguistic approaches to machine translation

2.2.1. Interlingua Based Translation

In interlingua-based translation a sentence is translated by first translating it into an interlingua, then into a sentence in the target language. This approach is comparable to translation using a pivot language, whereby translation is achieved by first translating a sentence into a pivot language, such as English, then translating the resulting English sentence into the required target language. This can be useful if resources for training a translation system for rarely used languages pairs, such as Maltese to Mongolian, are scarce, while resources for the respective translations to and from the pivot language have higher availability. However in contrast to a natural language, an interlingua is a language, which stores only the pure meaning of the source sentence, while adding no new translation errors through ambiguities and the like. Such an interlingua is not necessarily human understandable — a sentence in an interlingua could have the form of a vector $v \in \mathbb{R}^n$. While abstracting from the source sentence, all relevant information in the sentence is stored in the interlingua representation. This also includes such information as mood, politeness level, usage of active or passive voice or targets of actions.

As a result, two sentences in different languages with the exact same meaning result in the same interlingua representation, therefore also resulting in the same target sentence, when translated into the target language. This allows for multilingual translation without ever having to explicitly adjust the translation system to a particular language pair. Furthermore, in order to enable translation between n languages only one translation system containing n modules for translation into the interlingua and another n modules for translation from the interlingua is needed, instead of $n(n-1)$ separate translation systems.

2.2.2. Word Representations

As the relevant NMT systems use RNNs, the network input is partitioned into timesteps, taking one word from the source sequence at a time. Since neural networks operate with numerical input, it is essential to choose an appropriate representation of those words.

Word Embeddings One possible representation of words can be obtained by using a technique called word embedding (Mikolov et al., 2013a), in which a word is projected into a continuous vector space $\mathbb{R}^n$. Building a vocabulary of fixed size m, containing every possible word, with a fixed index for each word, a word is then presented to the neural network in the form of a one-hot vector. This one-hot vector is an m-dimensional vector with a one-valued entry in the dimension corresponding to the word index, and zero-valued entries in every other dimension. A word in the input sequence w_i in a one-hot representation is then projected into an embedding space, by multiplying it with the embedding matrix $E \in \mathbb{R}^{n \times m}$,

$$x_i = E \cdot w_i$$

This embedding matrix E is obtained via backpropagation, in the course of training the neural network that uses these word embeddings.
This embedding space has the desirable characteristic, that words with similar meanings are closer to each other, than other words. Furthermore, the embedding space has additive properties, such as:

$$emb(\text{king}) - emb(\text{man}) \approx emb(\text{queen}) - emb(\text{woman})$$

where *emb* is the projection of a word into the embedding space.

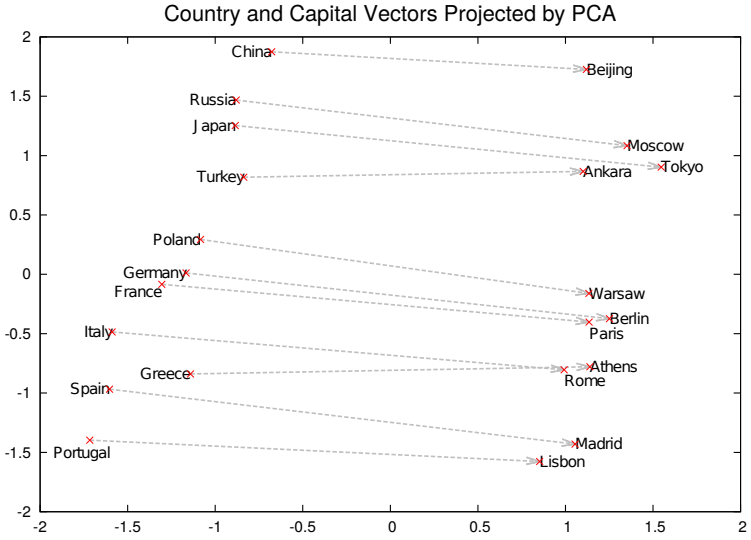

Figure 2.7.: Two-dimensional PCA projection of trained word embeddings illustrates how a model by itself organizes concepts and implicitly learns the relationships between them (Mikolov et al., 2013b)

Output Encoding The network output is encoded with an m-neuron softmax-layer, representing the probability for the corresponding word in the target vocabulary in each neuron. Since words in the input sequence and the output sequence potentially come from different vocabularies, as is the case in translation, we differentiate between source and target vocabulary.

Byte Pair Encoding Sentences in natural language consist of words, which in turn consist of characters. The NMT system reads a source sentence word by word and then generates a translation. Those words, however, are not necessarily words as they are known in natural languages: The sentences can be partitioned on a word-, subword- or character-basis.

The use of whole words as translation units results in large vocabularies, often more than 100,000 words. As the amount of neurons in input and output layers corresponds to the size of their respective vocabularies, this greatly increases the number of network parameters. Furthermore, *out of vocabulary* words (OOV) cannot be interpreted by the network and are replaced with a special *UNK* symbol during translation. This greatly reduces translation quality, and furthermore the NMT system becomes unable to cope with the fact that vocabularies in natural languages grow with time.
Using characters as translation units instead of words reduces the vocabulary size to the size of the used alphabets. Unlike with word based translation, the NMT system also has the ability to assign meanings to words, it has never seen during training by assigning meanings to parts of the word. By using appropriate training data, the network can also learn to recognize and correct spelling mistakes. The character-based approach, however, greatly increases the sequence lengths, resulting in a very deep network when unfolded for backpropagation through time, thus making it more difficult to learn due to resulting long term dependencies.
As the middle ground between these two approaches exists subword-based translation. With a technique called *Byte Pair Encoding* (BPE, Gage, 1994; Sennrich et al., 2015), words are split into subwords, which greatly reduces the vocabulary size, while also reducing the number of OOV words.

2.2.3. Encoder-Decoder Architecture

In Neural Machine Translation (NMT) the probability of a sentence $e = (e_1, ..., e_m)$ being the translation of a source sentence $f = (f_1, ..., f_n)$ is expressed as the conditional probability

$$\mathbb{P}(e|f) = \mathbb{P}(e_1, \ldots, e_m|f) = \prod_{i=1}^{m} \mathbb{P}(e_i|e_1, \ldots, e_{i-1}, f) \tag{2.7}$$

This means that the target sentence is generated one word at a time, while factoring in the source sentence and the previously generated target words. RNNs are especially suitable for this task, as they inherently calculate $\mathbb{P}(x_t|x_1, \ldots, x_{t-1})$ for the timestep t and an input sequence $(x_1, \ldots, x_n)$. As we also generally have variable length inputs, RNNs provide a further advantage.

As stated by Bahdanau et al. (2014) most of the proposed neural machine translation models belong to a family of encoder-decoders. As the name suggests, it consists of an encoder and a decoder, which in turn consist of RNNs. The encoder reads the whole input sequence and produces an encoded representation of the source sequence. In the first encoder-decoder models, as proposed by Cho et al. (2014b), this representation, called the summary of the input sequence, is the encoder RNN final hidden state. The summary is thus a fixed-length vector in a continuous vector space.

From the summary the decoder subsequently generates word by word the target sequence, by taking the summary, the previously generated word and the previous hidden state as an input. Using a softmax-layer as the decoder output, the decoder is trained to calculate the target word probabilities as shown in equation 2.7, representing the target word as described in section 2.2.2.

2.2.4. Attention-Based Translation

Word Alignment

When translating a sentence from one language into another, one word can often correspond to a word in the other sentence. However, a direct alignment of the i-th word in one sentence to the i-th word of the other is rarely the case, since different languages usually have different word orders. A clear one-to-one

alignment of words is also often not possible, as groups of words can often result in a single word after translation or the other way round.

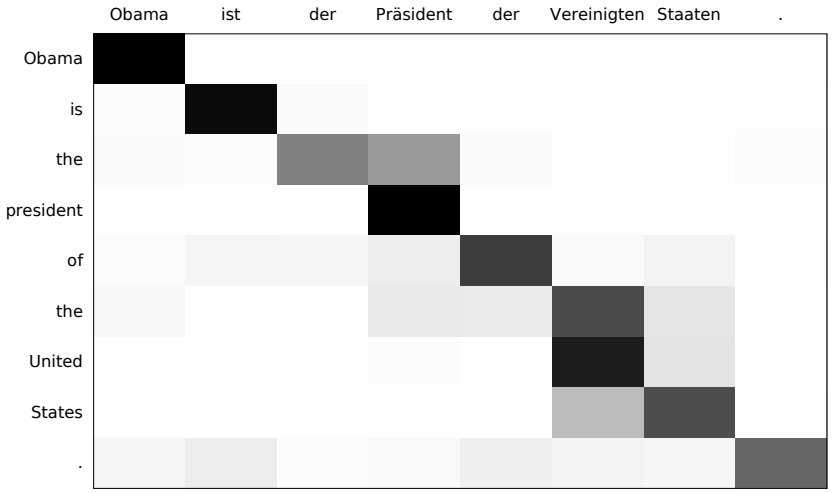

Figure 2.8.: Exemplary word alignment represented as alignment matrix (generated by one of the NMT systems trained for our experiments)

Attention

In combination with such advanced techniques, as beam search for decoding, the simple encoder-decoder translation system produces considerably good results. However, due the fact that the entire source sentence is encoded into a single fixed size vector, translation quality worsens with the increasing sentence length, as more information has to be encoded into this vector. To address this problem Bahdanau et al. (2014) introduces into this architecture an attention mechanism, extending the single vector provided to the decoder into a set of vectors.
Given a source sequence $(x_1,\ldots,x_n)$ the encoder produces a sequence of hidden states $(h_1,\ldots,h_n)$, known as *annotation vectors*. Each of the annotation vector h_j summarizes the corresponding word in the source sequence x_j and its context. In the decoding process, for each target word y_i a soft alignment α_{ij} is generated, describing a word alignment towards each word in the source sequence x_j. The alignment is generated from a normalized measure of similarity a between the each annotation h_j and the decoder RNN hidden state s_{i-1}. Using those alignments as weights, a *context vector* c_i is then calculated as the weighted sum of the annotation vectors:

$$\begin{aligned} e_{ij} &= a(s_{i-1}, h_j) \\ \alpha_{ij} &= \frac{exp(e_{ij})}{\sum_{i=1}^{n} exp(e_{ij})} \\ c_i &= \sum_{j=1}^{n} \alpha_{ij} h_j \end{aligned} \tag{2.8}$$

The alignment determines how much attention is assigned to each word from in source sequence in each decoding step, hence the name attention mechanism. The similarity measure a can be obtained in various ways, such as by the cosine similarity, or from a simple feed forward network.

In this attention model, instead of the fixed-length summary of the source sequence, the decoder now receives the context vector c_i in each decoding step.

Additionally, in order for the annotation to not only summarize the preceding words, but also the following words, Bahdanau et al. (2014) use a bidirectional RNN (BiRNN, Schuster et al., 1997) in the encoder. This BiRNN f reads the source sequence forwards, as well as backwards and produces the annotation vector for each word from the concatenation of both resulting hidden states:

$$\begin{aligned} \overrightarrow{h_i} &= f(x_i, \overrightarrow{h}_{i-1}) \\ \overleftarrow{h_i} &= f(x_i, \overleftarrow{h}_{i+1}) \\ h_i &= [\overrightarrow{h_i}, \overleftarrow{h_i}] \end{aligned} \tag{2.9}$$

2.2.5. Training the NMT System

As the translation system consists of multiple differentiable blocks, according to the rules of differentiation the whole translation system is also differentiable. This allows for the translation system to be trained in a end-to-end fashion, with the use of backpropagation on whole system. This also means that if an alignment model which is trainable via backpropagation, such as an FFNN was used, then it can then also be trained jointly together with the rest of the system.
This system of neural networks, which are known to require great amounts of training data, is trained by using the so called *parallel corpora*. Parallel corpora are sequences of sentences and their respective translations into another language, which are then used in supervised learning. In the decoding step t the system is trained to maximize the conditional likelihood $\mathbb{P}(y_t|y_1,\ldots,y_{t-1},x)$ for the source x and the reference $y = (y_1,\ldots,y_m)$, by minimizing the cross entropy error with backpropagation through time, using a one-hot encoding of the target word y_t as label.

2.2.6. BLEU

In order to evaluate the quality of a machine translation system, we require a way of evaluating the translations produced by it. As human evaluation would be highly time and cost inefficient for the amounts of data that needs to be evaluated, we need a way of automating this task. The criteria for the evaluation of translation quality are its adequacy and fluency.

One metric for translation quality measuring those criteria is BLEU (bilingual evaluation understudy). Given a translation and a human-generated reference sentence, it calculates a score by comparing the n-gram overlap. An n-gram is a sequence of n contiguous words. On a document level (as opposed to sentence level) BLEU then calculates the amount of n-gram overlap, relative to the total amount of n-grams for $n = 1,\ldots,4$. The final BLEU score is calculated from the geometric mean of the resulting values

$$\begin{aligned} \text{BLEU-}n &= BP \cdot \frac{\#n\text{-gram overlap}}{\#n\text{-grams}} \\ BLEU &= 100 \cdot \sqrt[4]{\prod_{n=1}^{4} \text{BLEU-}n} \end{aligned} \tag{2.10}$$

BP is the *brevity penalty*, which penalizes short translations, as they tend to get higher scores. BLEU tries to lean towards adequacy for small values of n and towards fluency for higher values of n and find the common ground between them.

Despite frequent criticism as there are many issues with BLEU, it is still the most commonly used evaluation metric, as it has been found to be the best available approximation of professional evaluation, while still having an acceptable (polynomial) computation complexity.

3. Related Work

3.1. Multilingual Attention-Based Translation

Based on the attention-based encoder-decoder architecture described by Bahdanau et al. (2014), Ha et al. (2016) have proposed an approach to extend this architecture to multilingual translation. The idea is to train the NMT system using a unified vocabulary and training corpus across all languages, while making no modifications to the architecture. For this, Ha et al. (2016) describe two techniques in the form of input pre-processing steps:

- **Language-specific Coding** words of different languages are distinguished through language codes. This, for example, can look like this: *bank* → *@**en**@bank*
- **Target Enforcing** A symbol indicating the desired target language of the translation is added to the beginning and at the end of the sentence.

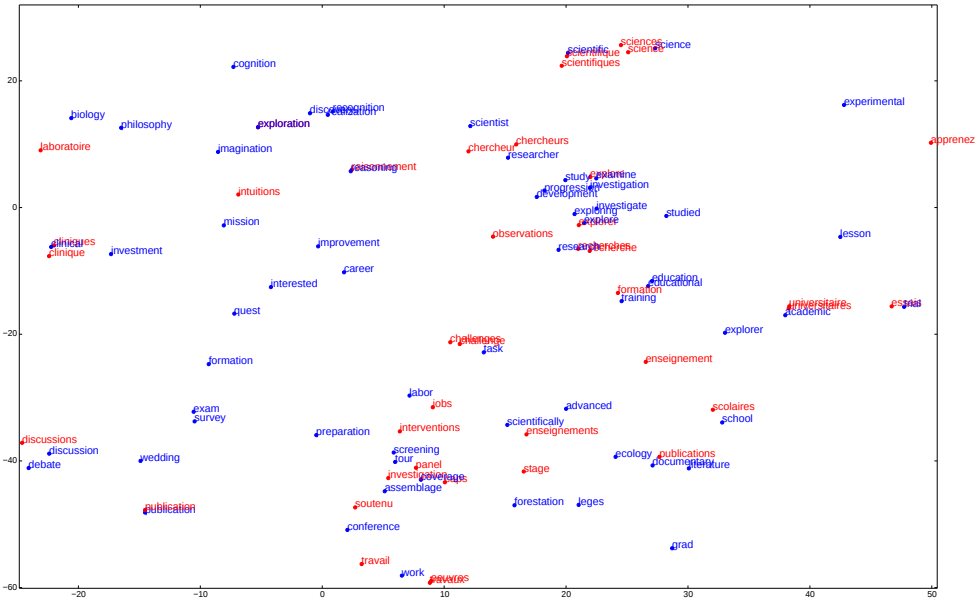

Figure 3.1.: Visualization of the shared embedding space via dimensionality reduction (Ha et al., 2016).

Shared Embedding The approach of using shared vocabularies across multiple languages also results in a shared embedding space. As Ha et al. (2016) have shown in their experiments, the NMT system learns to correlate words of different languages in this shared embedding space in a helpful way: as seen in figure 3.1 words with similar meanings from different languages end up closer to each other. As an

example in figure 3.1 we can observe the English words *scientific* and *science* clustered together with their French translations.

The goals of this approach are to

- improve translation quality for individual languages, by letting the NMT system learn common semantics across languages, thus helping the system to better generalize
- improve translation quality for language pairs, for which parallel training data are scarce, and in the extreme case even allowing for zero-shot translation, by letting the NMT system find a representation of the sentences, which abstracts from the individual languages
- reduce the amount of translation systems needed for translation between n languages from $n(n-1)$ to a single one, thus reducing the amount of training time and the amount of parameters

3.2. Examination of Neural Sentence Representations

Prior to the introduction of attention to the encoder-decoder architecture (see section 2.2.3) Cho et al. (2014b), Sutskever et al. (2014) and Shi et al. (2016) among others have examined the encoder sentence representation. The former two have explored the ability of the encoder to represent sentences at the level of their meaning by comparing the relative positions of sentences close to each other in terms of meaning. In their visualization of selected few sentences by means of a 2-dimensional PCA projection (see figure 3.2) Sutskever et al. (2014) show a discernible additive relation between sentence representations, closely resembling the relation between words in word embeddings. This indicates that the encoder in their NMT system does indeed have the capability of abstracting from language and representing the translated sentence at the level of their meaning.

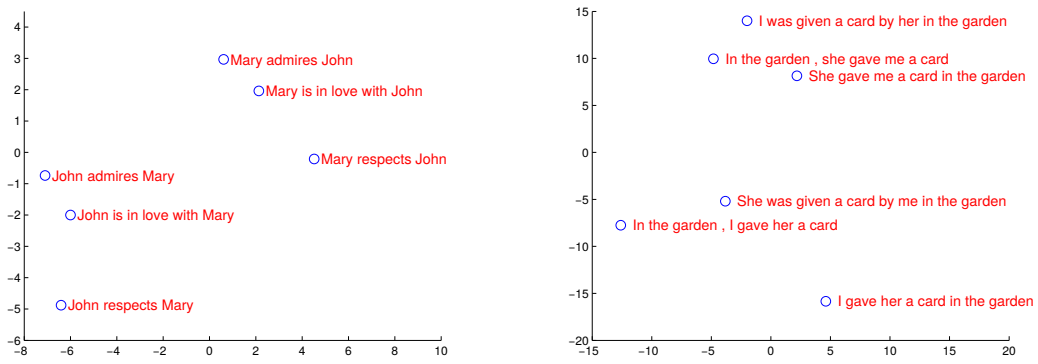

Figure 3.2.: Visualization of the sentence representations via dimensionality reduction (Sutskever et al., 2014).

Shi et al. (2016) have examined the sentence representation on a syntactic level and have found that the encoder implicitly learns to store information about the source sentence syntax in the sentence representation.

This sentence level representation, which Cho et al. (2014b) call *summary* is the equivalent to the context vectors in the attention-based model, with the difference being that context vectors represent only the part of the sentence it puts attention to.

3.3. Examination of Context Vectors in the GNMT System

Based on the same principle as Ha et al.'s (2016) multilingual NMT system, Johnson et al. (2016) have proposed a multilingual attention-based encoder-decoder NMT system. In the course of their experiments Johnson et al. (2016) have examined the hidden representation of the translated sentences in their NMT system in respect to its resemblance of an interlingua representation. This hidden representation they call the *attention vectors*, and is equivalent to what we call context vectors in this thesis. Using a t-SNE projection into three-dimensional space, Johnson et al. (2016) observe that attention vectors for semantically identical sentences form clusters. Thus they visually confirm, that their NMT system learns to organize sentence representation by their meaning, which they call "early evidence of shared semantic representations (interlingua) between languages". Furthermore Johnson et al. (2016) have found a correlation between the translation quality for these semantically identical sentences of different languages and the similarity between the attention vectors for these sentences.

3.4. Generative Adversarial Networks

Generative Adversarial Networks (GAN, Goodfellow et al., 2014) are a type of neural network, used in an approach for training generative models in an unsupervised fashion. It consists of a generative network G, which tries to generate data and a discriminative network D, which tries to differentiate between data generated by G and the training data. G is then trained to "trick D into thinking" that the data generated by G originates from the training data by maximizing the error for D. In this adversarial manner G is trained to produce data which is indistinguishable from the actual training data.

Goodfellow et al. (2014) compare G to counterfeiters, trying to produce fake currency and D to the police, which tries to detect fake currency; in a zero-sum game they iteratively improve each other, until the fake currency is indistinguishable from real currency.

The approach of letting a discriminating network D classify the output of another network G is similar to the procedure used in this thesis: we build a discriminator D on top of the attention mechanism of a NMT system G, while ideally looking for D to fail in its classification task. Unlike with the approach with GANs however, we do not take the next step of adjusting G to maximize the error for D. We describe the potential future work on this matter in section 6.1.

3.5. IND-CPA Security

In cryptography for an encryption scheme exists the property of IND-CPA security, which is equivalent to semantic security. IND-CPA (*indistinguishability under chosen-plain text attack*, Goldwasser and Micali, 1982) security is defined via the IND-CPA game, which informally is described as:

For an encryption algorithm Enc and an adversary A, after A chooses two plain text messages $M_1 \neq M_2$, A receives the encryption C^* for one of those messages (randomly chosen). A wins the game if A succeeds in assigning the chiffre C^* to the message it originated from.

Enc is IND-CPA secure if every possible adversary A only succeeds in winning this game with a probability $\mathbb{P}(A \text{ wins}) = \frac{1}{2} + \varepsilon$, whereas ε is a negligible term[1]. In this case A is said to only have negligible advantage over random guessing.

This procedure is in its essence similar to the procedure we use in this thesis: in a multilingual NMT system which perfectly learns a language independent representation, we believe the context vectors to be indistinguishable under a chosen language pair. We thus simulate a polynomial time adversary A, which tries to correlate a given context vector c^* to a language pair using neural network based classification,

[1] In a formal definition there are further details to notice, such as A is a polynomial time adversary and ε is negligible in a security parameter k

hoping that A fails in this task. However, unlike with IND-CPA security we can only prove an absence of perfectly independent representation with this method, since we only look at a specific adversary A.

4. Examination of Context Vectors

4.1. Motivation

Prior to the introduction of attention to encoder-decoder NMT systems, a source sentence read by the encoder was encoded into a fixed length vector, forcing the encoder to find a meaningful sentence representation, storing only its meaning. With attention this meaningful representation has moved from this single fixed length vector to a set of multiple vectors, the so called context vectors; the principle however stays the same. Adding multiple languages to the source and target side of the encoder and decoder increases the problem complexity while keeping the amount of parameters constant, compelling the network to generalize by using shared semantics between languages. Under such circumstances the NMT system would ideally learn a sentence representation, which stores only its meaning, while abstracting from the source and target languages.

This principle of translating a sentence into its language-independent meaning is known in linguistics as an interlingua representation, and the idea has been known for many centuries. However, such an interlingua has presently not yet been arrived at.

As the context vectors are strongly reminiscent of such an interlingua representation, this bears the question of just how close it is to a true interlingua. In other words, we want to know how well the NMT system learns to abstract from the languages involved in translation, and only store the pure meaning of the sentence to be translated in its hidden representation.

One way of evaluating how well the NMT system abstracts from language would be to look at the independence of this hidden representation from the concrete language pair involved in its generation. Assuming that the NMT system perfectly learns to extract the meaning from a sentence, sentences of different languages would arrive at the same representation and would thus be indistinguishable in terms of the languages involved. In this thesis we shall examine this particular question, whether the context vectors in the proposed multilingual NMT system are independent of the language pair.

4.2. Approach

Considering the fact that showing the independence of the context vectors from the source and target language would require a formal proof, we shall approach the problem by studying the dependency, which, if present, can be discovered through experimental means. Given a context vector, we shall try to identify the language pair it was generated from, and declare the dependence in the case of success. As this problem can be formulated as finding a correlation between a vector $c \in \mathbb{R}^n$ and one language from a set of candidates $\{l_1, \ldots, l_k\}$, this calls for classification.

Given the recent success of neural networks in discriminative tasks with high dimensional input, we shall approach this particular classification problem with classification via neural networks. With the fixed length of the input vectors and no reason to believe shift-invariant features to be present in the context vectors, we believe a simple feed forward neural network with fully connected layers to be the most appropriate type of network as the basis for the classifier.

Using supervised learning, we shall train such a classifier, providing context vectors and their respective true source-target language pair as the label. The data used for this learning task will be generated by first training an attention-based encoder-decoder NMT system as described by Bahdanau et al. (2014), extended to multilingual many-to-many translation as described by Ha et al. (2016). We will subsequently extract the context vectors while sampling translations of the validation data.

4.2.1. Neural Classification

We will try to predict the correct class for a given context vector by using a FFNN with fully connected layers. Starting with the simplest approach we will first attempt linear classification using a network without any hidden layers (SLP), which results in an output layer directly connected to the input layer. We will call this *linear classification*. The capability of such a network to successfully detect the presence of a dependence would imply a linear separability of the context vectors, allowing the network to partition the feature space into relevant classes.
After the linear classification we will attempt a classification using an MLP-classifier with one or more hidden layers, to look for nonlinear features.

The labels will be encoded as concatenation of two one-hot vectors, the first vector encoding the source language and the second vector encoding the target language. The classifiers will then be trained using a softmax-layer as the output layer and cross entropy error between the first and second half of the network output and labels:

$$o_s = \text{softmax}(D(x)_{1...5}) \qquad o_t = \text{softmax}(D(x)_{6...10})$$
$$t_s = t_{1...5} \qquad t_t = t_{6...10}$$
$$E_s = H(o_s, t_s) \qquad E_t = H(o_t, t_t)$$

$$E = \frac{E_s + E_t}{2}$$

for the classifier D and the input-label pair (x,t). The network is then trained to minimize E using *adam* (Kingma and Ba, 2014) as optimizer. The classifier predicts the language pair using the *argmax* of the output first and second half:

$$(L_s, L_t) = (argmax(D(x)_{1...5}), argmax(D(x)_{6...10}))$$

4.2.2. Investigating Linear Relation of Context Vectors

The results of our experiments have shown that a linear classifier is capable of correctly classifying the context vectors. As described in section 4.2.1, this implies the linear separability of the context vectors. Suspecting the existence of a linear translation, such that a context vector of one language can be obtained, when applied to a context vector of another language (similar to the additive relation between words with word embeddings, described in section 2.2.2), we decided to further investigate this matter.

Taking context vectors x with s_1 as their source language and t_1 as their target language, and another language s_2 we will try to find a vector b, such that $x+b$ is recognized as a context vector with s_2 as its source language. In order to find such a vector b for the context vector x we will need a comparable context vector x' which has s_2 as its source language. To obtain this counterpart x' for x, we need to ensure that two matching source sentences f_1 and f_2 in our parallel corpus with s_1 and s_2 as their respective language, are both translated into the same target sentence in the language t_2, allowing for the direct comparison of the generated context vectors. For this we will adjust the NMT system sampling mechanism to accept a reference target sentence $e = (e_1, \ldots, e_m)$ and use e_{t-1} as the input for the decoding step t, instead of the previously generated target word y_{t-1}. With this method we will obtain a pair of context vectors (x_t, x'_t) for each target word e_t, which can then be used as an input and its label in the training of b.

Using a training set with German-English context vectors for the input, and matching Dutch-English context vectors as labels, we will again use gradient descent to train a translation by randomly initializing a vector b and then minimizing the *summed squared error*

$$sse(x+b,t) = \frac{1}{2}\sum_i (x_i + b_i - t_i)^2$$

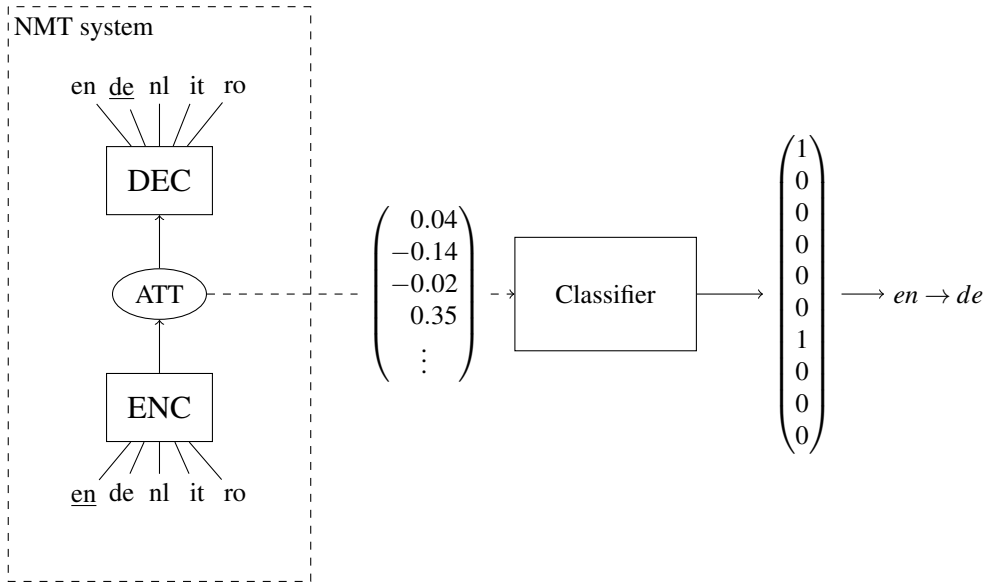

Figure 4.1.: Schematic description of the procedure we use to test for the dependency of the context vectors. During translation the context vectors are extracted from the attention module, and fed to the classifier. The classifier output is a "two-hot" vector representing the predicted source and target language in the its first and second half (every language is assigned a fixed dimension, e.g. English in dimension 0,...).

for each input-label pair (x,t).

The resulting translated vectors $x+b$ will then be fed to the previously trained classifier, in order to see, whether they are recognized as Dutch-English context vectors. Furthermore, in order to see how this translation affects context vectors of different target languages, we will apply this translation b to German-Italian context vectors. We expect the resulting vectors to be classified as Dutch-Italian context vectors.

5. Evaluation

5.1. Multilingual Translation Models

In order to examine the context vectors we have trained translation models as described by Ha et al. (2016).

Training Data For training we have used the multilingual WIT[3] (Cettolo et al., 2012) training corpus, which provides high quality multilingual translations. This corpus consists of transcriptions of 200,000 English sentences from TED Talks, and their translations into German, Dutch, Italian and Romanian. As the evaluation of the NMT system ability to provide translations in an under-resourced training scenario is not the purpose of this thesis, we have trained the NMT system using every possible combination of source and target languages, except for language pairs where the source and target language is the same. This gives us 20 language pairs, and therefore 20 valid classes of context vectors and a total of 4,347,886 sentence pairs.

For the purpose of evaluation we have used another development data (devdata) set consisting of 900 sentences from TED Talks involving the same five languages.

Translation System The translation models have been trained using Nematus (Sennrich et al., 2017), which provides an attention-based encoder-decoder NMT system as proposed by Bahdanau et al. (2014), implemented in Python using Theano (Theano Development Team, 2016). Nematus uses a special type of RNN, which Sennrich et al. (2017) call *conditional GRU with attention* in its decoder. As alignment model Nematus uses a feed-forward *tanh*-layer, which is jointly trained together with the rest of the system. We have used subword translation units.

In order to achieve multilinguality as described by Ha et al. (2016), all 20 parallel corpora have been merged into one multilingual parallel corpus through the concatenation of the source and target texts in the same respective order. The source and target vocabularies have been built from the merged corpus after applying every pre-processing step. The pre-processing steps include:

1. tokenization: space-separation of translation symbols such as punctuation from each other
2. true casing: changes capitalization of words at the beginning of sentences according to their normal capitalization
3. byte pair encoding
4. language specific encoding
5. target enforcing

With German as target language, applying those pre-processing steps to the sentence

```
This is an English sentence andareallylongword.
```

thus results in the sentence

```
@de @en@this @en@is @en@an @en@English @en@sentence @en@and@@ @en@are@@
@en@all@@ @en@y@@ @en@long@@ @en@word .
```

As we assume, that a smaller hidden layer size will force the network to abstract from the source language even more, we have trained models with different sizes of hidden layers in order to test wether this holds true. Furthermore, in order to compare how classification results evolve with the ongoing training of the translation system, we also evaluate one particular model at different checkpoints in the course of training.

Model Parameters For the evaluation task we have trained three translation models. We trained all models using the same setup and network parameters, differing only in the amount of training time and the hidden layer sizes, which is 1024 for the first model, and 512 for the second and third models. We used English, German, Dutch, Italian and Romanian as source as well as target languages, whereby words were pre-processed into subword units with BPE[1], using a BPE merging parameter of 39,500 trained on the merged corpus before applying language specific encoding, resulting in a total vocabulary size of 88,000 words. The first two symbols in vocabulary are the end of sentence marker and *UNK*: every word not contained in the fixed vocabulary is replaced with *UNK*. We used a maximum target sentence length to 50 and a word embedding size of 500.

Training We trained the NMT system using the merged training corpus described in section 5.1 containing a total of 4,347,886 sentence pairs, and evaluated it on a separate devdata set, which like our training data also consists of transcriptions of TED Talks. We have trained all models using a batch size of 40, *adam* as optimizer and dropout training (Srivastava et al., 2014), with a dropout ratio of 0.2 for the embedding layers and hidden layers and 0.1 for the source and target layers. Since we use a concatenation of the bidirectional encoder forward and backward hidden states as annotation vectors, this resulted in context vectors of size 2048 for the first model and 1024 for the second and third models.

The first model, with the hidden layer size of 1024 was trained for 5 days on a NVIDIA GTX 1080Ti GPU, coming to an early stop after 110,000 iterations on the 5th day. As shown in table 5.1, this resulted in a final BLEU score of 11.94 on the merged devdata set.
Another model, with a hidden layer size of 512 was trained for 5 days on a NVIDIA GTX 1080Ti GPU, resulting in the second model and another 5 days on a NVIDIA Tesla K20m, resulting in the third model after a total of 260,000 iterations. As shown in table 5.2 and 5.3, this model achieved a BLEU score of 11.07 after training for 5 days and 14.94 after training for 10 days.
All translations used for calculating BLEU scores were generated using beam search decoding, with beam size 5.

[1] for BPE we used the script from the subword-nmt repository (`https://github.com/rsennrich/subword-nmt`). For tokenization and true casing we used the scripts provided by the Moses framework (`http://www.statmt.org/moses/`)

src \ trg	en	de	nl	it	ro	avg
en		13.78	13.19	13.68	11.20	12.96
de	19.43		11.31	9.84	6.98	11.64
nl	17.30	10.38		9.42	6.67	10.94
it	19.04	9.85	10.31		8.28	11.87
ro	18.57	9.27	9.71	10.68		12.05
avg	18.33	10.82	11.13	10.90	8.28	11.94

Table 5.1.: BLEU scores for trained NMT system with hidden layer size 1024

src \ trg	en	de	nl	it	ro	avg
en		13.24	12.84	12.71	9.39	12.04
de	18.14		10.91	8.79	5.94	10.94
nl	16.04	9.34		8.43	6.26	10.01
it	17.64	9.60	9.91		7.62	11.19
ro	16.72	8.82	9.37	9.62		11.13
avg	17.13	10.25	10.75	9.88	7.30	11.07

Table 5.2.: BLEU scores for trained NMT system with hidden layer size 512 after 160,000 training iterations

src \ trg	en	de	nl	it	ro	avg
en		17.82	16.17	17.27	14.75	16.50
de	23.21		13.84	12.07	9.65	14.69
nl	20.42	12.46		12.11	9.44	13.61
it	22.84	12.12	12.35		11.16	14.62
ro	22.85	11.87	12.23	14.10		15.26
avg	22.33	13.56	13.64	13.88	11.25	14.94

Table 5.3.: BLEU scores for the same NMT system with hidden layer size 512 after 260,000 training iterations

From the the average scores for the source and target languages we observe considerable variation in the average translation quality between different target languages, the difference between English and Romanian as target language being 11.08 BLEU points for the third model. With the greatest difference between different source languages being 2.89 BLEU points, the source language had a comparatively low impact on translation quality.

However, this result is intuitive, as the problem of language generation is harder than the analysis.

5.2. Classification of Language Pairs

Using the devdata as translation source, we generated and extracted the context vectors and the correct language pairs from *Nematus* using the previously trained models. This resulted in 459,878 context vectors for the first model, 457,054 vectors for the second model, and 444,570 vectors for the third model, with overall 20 classes. The number of context vectors matches the number of translation symbols in the generated target sentence, and thus differs for each translation model, since they produce different translations. We merged the data from each class into one sequence by alternating between single sentences of each class, ensuring that each mini batch of size 1000 to contained samples of all 20 classes, considering the sentence maximum length of 50.

Using the *TensorFlow* (Abadi et al., 2016) low level API, we built for each model SLP-classifiers, and MLP-classifiers with one hidden layer containing 64 hidden neurons. Having tried out various sizes of the hidden layer, we found a size around $\sqrt{n}$ to work best for n-dimensional context vectors, as higher amounts quickly result in overfitting. All classification accuracies were calculated as ratio of correctly predicted language pairs, using 25% of the context vectors as the validation set.

5.2.1. Results

As illustrated in table 5.4 all classifiers have achieved significantly high classification accuracies, with linear classification achieving 86-96% correct classification rates after 50 epochs of training, and slightly higher rates for their nonlinear counterparts. These values are also representative for the classification rates of the source languages alone, as the target languages have been correctly classified with near 100% accuracy by all the classifiers. Having tested *sigmoid*, *tanh* and *ReLU* for the hidden layer activation functions, we have found only minimal differences in resulting accuracies, with *ReLU* performing the best by small margins of maximum 1%. As seen in figure 5.1 and 5.2 high classification rates were already achieved after the first epoch, requiring only several seconds of training on an NVIDIA Tesla K20m GPU.

model	linear classification	MLP-classification
big HL	95.75%	96.09%
small HL	85.93%	89.94%
small HL, more training	91.93%	94.29%

Table 5.4.: Classification rate with zero and one hidden layers after 50 epochs of training. *big HL* and *small HL* here refer to the hidden layer size of the NMT model used for generating the context vectors.

These results strongly suggest that the extracted context vectors are not independent of the source language. The high classification rates of the linear classifiers further suggest a linear relation between context vectors of different languages. In view of the fact, that the classification rates for the linear classifier increase with ongoing training of the NMT system, it is apparent that the linear features which the classifier makes use of become more distinctive with the progression of the training.

The confusion matrix (see table 5.5 and figure A.1, A.2) shows a discernible correlation between the language specific BLEU scores and classification errors for that language, Romanian being the language with lowest BLEU scores, as well as with most classification errors. Furthermore, there is also a noticeable correlation between language similarity and the classifiers tendency to confuse them with each other, as can be seen with Dutch being commonly misclassified as German and Romanian misclassified as Italian.

For the classification with the MLP we can observe slight increases in classification rates. However, for the context vectors generated by the better trained NMT systems, which we consider more interesting,

these increases are insignificant. This makes it difficult to draw conclusions about the presence of purely nonlinear features, as the classifiers mostly make use of the linear features.

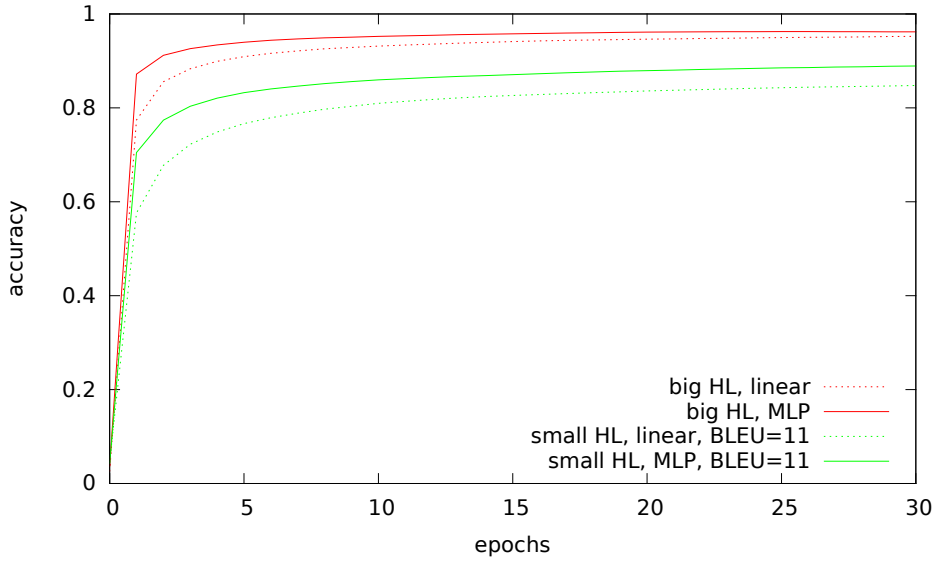

Figure 5.1.: Comparison of the training curves for the classifiers based on the translation model with a big hidden layer size (red) and small hidden layer size (green). The curves show that a smaller hidden layer size makes the classification task more difficult despite similar translation quality.

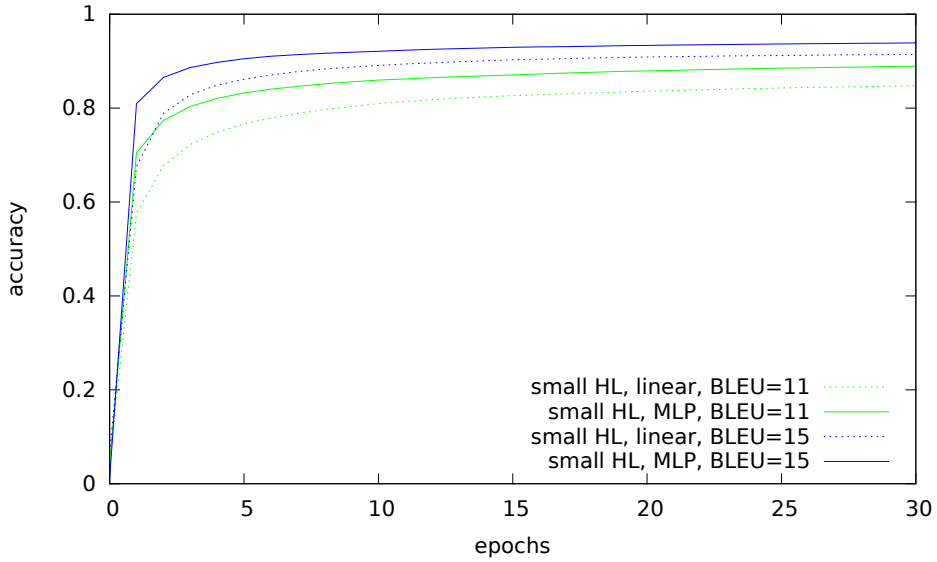

Figure 5.2.: Comparison of the training curves for the classifiers based on the translation model with a small hidden layer size and higher BLEU scores (blue) and lower BLEU scores (green). The curves show that with the progress of training the classification task becomes easier.

pred \ label	en	de	nl	it	ro
en	22398	25	21	15	42
de	32	21818	129	19	24
nl	24	326	24784	43	89
it	171	46	74	19728	2490
ro	85	54	94	628	20841

Table 5.5.: The Confusion matrix shows a comparison between predicted source languages and labels

5.3. Linear Relation between Context Vectors

To investigate the supposed linear relation between the context vectors of different languages, we successfully trained a translation as described in section 4.2.2. For this we first modified *Nematus* to accept a reference target sentence for translating a source sentence.

Using the German and Dutch translation of the devdata set as translation source and the English translation as the reference, we generated a training set with the German-English context vectors as the input and the Dutch-English context vectors as the labels, using the third translation model. Using *adam* as optimizer we then trained a vector b, by minimizing the *summed squared error*. The training of such a translation for 20 epochs resulted in a vector b with a norm of 0.710 and a mean distance of 6.912 between the translations and their labels, the mean distance between untranslated German-English vectors and their Dutch-English counterparts being 6.939 (see figure 5.3).

Applying this trained translation to a validation set unseen in training, we further classified the originally German-English context vectors with the help of the previously trained linear classifier. As seen in table 5.6, 88% of the translated vectors were classified as Dutch-English. Furthermore, the application of this translation to German-Italian context vectors, resulted in 86% of these to be classified as Dutch-Italian (see table 5.7). Applying this same procedure for all 180 valid four-tuples of languages, from a total of 752,841 context vectors for 90,9% the source language was successfully translated as intended, while not affecting the predicted target language for these context vectors. For the original source language s, the translated source language s', the target language used in training the translation t, and the target language of the context vectors to which the trained translation was applied a four-tuple is considered valid if s, s', t are pairwise different and s, s', t' are pairwise different, resulting in 60 different translations, which are each applied to context vectors of 3 different target languages.

These results strongly suggest a fixed linear translation between context vectors of different languages, affecting only a few set dimensions of the context vectors.

source language	original	translation
en	0	0
de	1268	133
nl	100	1246
it	32	16
ro	13	18

Table 5.6.: Comparison of predicted source language for German to English context vectors, after training and applying translation of source language to Dutch

source language	original	translation
en	4	1
de	1233	172
nl	18	1081
it	0	0
ro	2	3

Table 5.7.: Comparison of predicted source language after applying the same translation to German to Italian context vectors

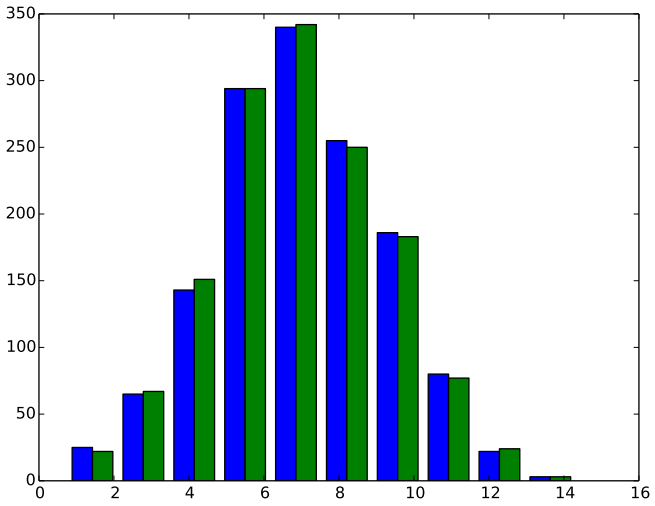

Figure 5.3.: The distribution of distances between original (blue) and translated (green) German to English context vectors to their correspondent Dutch to English context vectors shows, that the translation has negligible impact on the distances.

6. Conclusion

In this thesis we have explored the ability of the multilingual NMT system proposed by Ha et al. (2016) to produce a language independent representation for the sentence it translates. Having put forward the hypothesis, that with better abstraction ability of the NMT system the hidden representation of the sentence becomes more independent of the language pair that was involved in its generation, we have explored this independence by measuring the degree of correlation between the hidden representation and the language pair. Of the various hidden representations in the translation system we have chosen to examine the *context vectors*. In order to measure this correlation, we have built classifiers based on feed-forward neural networks which, given a context vector, attempt to predict the language pair involved in its generation. Having trained multiple multilingual NMT systems with 5 languages on the source side as well as the target side, we have trained single layer perceptrons to classify the context vectors produced by these systems. Those classifiers have achieved rates of correct classification of up to 95.75% for 25 possible classes, which, according to our hypothesis, indicates that our NMT system does not successfully abstract from the languages involved in the translation. Furthermore, having explored the underlying cause of success in classification, we have found present a linear relation in Euclidean space between context vectors of different languages. More precisely, for a pair of languages (A,B) we have found a vector b_{AB}, such that for a context vector c with A as its source language, $c+b_{AB}$ is classified as having B as the source language in 90.9% of the cases. We have found this translation for the source language to be independent of the target language.

However, we have found this translation to have negligible impact on the distance between the context vectors, and that the features, which the single layer perceptron makes use of for the classification become more distinguished with progressing training of the NMT system. In view of these facts, there exists the possibility that the linear relation does not affect the ability of the NMT system to bring sentences of different languages into a shared representation, in which sentences of similar semantic meaning end up closer to each other, and thus not compelling the NMT system to hide these features.

In order to find nonlinear features, have also trained multilayer perceptrons to accomplish the classification task. With those multilayer perceptrons we have only found marginal increases to classification rates over the single layer perceptrons, which makes it difficult to draw conclusions about the presence of purely nonlinear features.

In conclusion, we have found the sentence representation to not be language independent in our trained NMT systems. For this representation, however, it is surprisingly easy to transform the features representing the source language in a target language independent way.

6.1. Future Work

Finding Nonlinear Correlation Since for the classification task our MLP-classifier also made use of the linear features which we have found to be present, the question of to what extent purely nonlinear features, which help in this classification task are present has not been sufficiently answered. In order to test for purely nonlinear features, an MLP-classifier which cannot use the linear relation between context vectors could be trained, by eliminating these linear relations in the training set. These features could for instance be eliminated by using a modified training set, where each context vector c'_A is obtained by adding one of the found translations the the original context vector $c'_A = c_A + b_{AB}$ for a random language B.

Adversarial Training of NMT System As described in section 3.4, the approach used in this thesis is similar to the first stage in the procedure to train generative models with GANs. As GANs have

shown great success, the remaining steps of this procedure could be applied to training of the NMT system as well. By training the NMT system G to produce context vectors for which a discriminating network D is unable to predict the correct language pair in an adversarial manner, the NMT system could learn to produce indistinguishable context vectors. The NMT system would then be alternatingly be trained in supervised learning and adversarial unsupervised learning, potentially learning to abstract from languages.

Zero-Shot Translation The linear translation which we have found to be present between the context vectors could potentially be applied in order to improve zero-shot translation. Zero-shot translation is the task of producing translations for a language pair, which the NMT system has never seen during training. For a language pair (A,C), which the NMT system has never seen during training, and a language pair (B,C), which the NMT system has seen during training an attempt at improving translation quality for the unseen language pair could be made, by translating the context vectors c_{AC} to $c_{BC} = c_{AC} + b_{AB}$ for the previously found translation for source languages b_{AB}. This produces context vectors for a language pair which the NMT system is more familiar with.

A. Visualizations

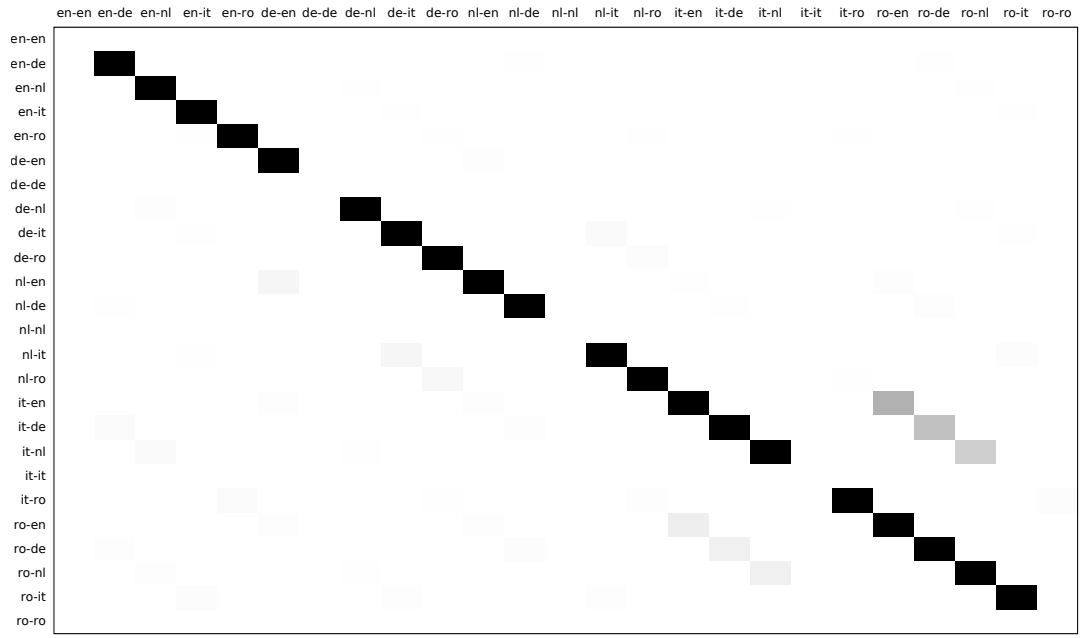

Figure A.1.: Comparison of predicted classes and labels represented as confusion matrix

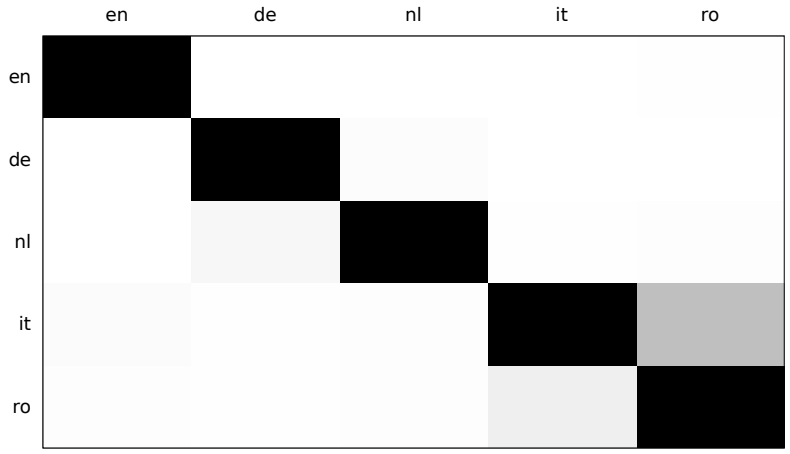

Figure A.2.: Same confusion matrix as in figure A.1 but summed by the source languages

Bibliography

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng
2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467. 24

Bahdanau, D., K. Cho, and Y. Bengio
2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473. 2, 11, 12, 13, 14, 18, 21

Cettolo, M., C. Girardi, and M. Federico
2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, Pp. 261–268, Trento, Italy. 21

Cho, K., B. van Merrienboer, D. Bahdanau, and Y. Bengio
2014a. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259. 8

Cho, K., B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio
2014b. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078. 11, 15

Gage, P.
1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38. 11

Goldwasser, S. and S. Micali
1982. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, Pp. 365–377, New York, NY, USA. ACM. 16

Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio
2014. Generative Adversarial Networks. *ArXiv e-prints*. 16

Ha, T., J. Niehues, and A. H. Waibel
2016. Toward multilingual neural machine translation with universal encoder and decoder. *CoRR*, abs/1611.04798. 5, 7, 2, 3, 14, 16, 18, 21, 28

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

Johnson, M., M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. B. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean
2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558. 16

Kingma, D. P. and J. Ba
2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980. 19

Mikolov, T., K. Chen, G. Corrado, and J. Dean
2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781. 10

Mikolov, T., I. Sutskever, K. Chen, G. Corrado, and J. Dean
2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546. 10

Schuster, M., K. K. Paliwal, and A. General
1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*. 13

Sennrich, R., O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A. V. M. Barone, J. Mokry, and M. Nadejde
2017. Nematus: a toolkit for neural machine translation. *CoRR*, abs/1703.04357. 21

Sennrich, R., B. Haddow, and A. Birch
2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909. 11

Shi, X., I. Padhi, and K. Knight
2016. Does string-based neural mt learn source syntax? In *Proceedings of EMNLP 2016*, Pp. 1526–1534. 15

Srivastava, N., G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov
2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958. 22

Sutskever, I., O. Vinyals, and Q. V. Le
2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215. 7, 15

Theano Development Team
2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688. 21

Waibel, A., T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang
1989. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339. 5