Universität des Saarlandes

Master's Thesis

Online Parallel Data Extraction with Neural Machine Translation

submitted in fulfillment of the degree requirements of the

MSc in Language Science and Technology at **Saarland University**

Author:
Dana Ruiter
Matriculation: 2565776

Supervisors:
Prof. Dr. Josef van Genabith
Dr. Cristina España i Bonet

May 13, 2019

Contents

EIDESSTATTLICHE ERKLÄRUNG

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

DECLARATION

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Saarbrücken, May 13, 2019

Dana Ruiter

Abstract

Neural Machine Translation (NMT) depends on large amounts of parallel data, which is scarce for low-resource language pairs and domains. Extracting parallel sentences from a non-parallel source using similarity measures over interlingual representations is our proposed method towards low-resource Machine Translation (MT). In an encoder-decoder NMT system, such representations can be observed for instance in the encoder outputs or the word embeddings when the model is trained on multilingual data. To exploit the fact that these representations improve in quality over time, this thesis project envisions to develop a parallel data extraction framework that extracts parallel data *online*, i.e., at the same time as the MT model is being trained. In this report, the literature review is presented, followed by an outline of the proposed system and the experiments performed.

Chapter 1

Introduction

Neural approaches to Machine Translation (MT) (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014b; Bahdanau et al., 2014; Luong et al., 2015a) have brought major advances in translation quality. To train such models, large amounts of sentence-aligned *parallel data* are needed, which is often scarce for low-resource language pairs or domains. As such, communities of low-resource languages can currently hardly be provided with high quality MT. Overcoming this constraint by either extracting parallel data from non-parallel sources or developing unsupervised techniques in NMT is therefore crucial to cover all languages and therewith also to the further propagation of freely accessible information as well as the breaking down of language barriers between language communities.

Non-parallel documents of similar contents, or *comparable corpora*, are becoming easier to obtain (Paramita et al., 2019), and extracting parallel sentences from them has recently become a wide research field. Some approaches focus on estimating similarities between Neural Machine Translation (NMT) internal representations to select pairs (Grégoire and Langlais, 2018; Artetxe and Schwenk, 2018; Bouamor and Sajjad, 2018). All of these approaches separate the parallel sentence extraction and NMT training tasks. However, in España-Bonet et al. (2017), it has been shown that in multilingual NMT systems, representations of sentences which are translations of each other tend to increase in cosine similarity as training progresses. That is, as training steps are performed, the NMT representations become better indicators of whether a sentence pair is parallel. In this project we want to exploit this fact by developing a joint extraction and NMT training system. The premise is that NMT systems —either sequence to sequence models with RNNs, transformers, or any architecture based on encoder-decoder models— learn strong enough representations of words and sentences to judge *online*, i.e. during NMT training, if an input sentence is useful. The stronger the internal representations, the more reliable pairs will be selected; and the better the pairs, the better the learning: a doubly *virtuous circle*.

The above being the motivation and main idea of our study, the following chapters will focus on a variety of aspects related to online parallel sentence extraction. After giving a brief introduction to MT, especially NMT and good practices that come with it, an overview of related research fields such as multilingual MT, domain adaptation and especially low-resource MT will be given. After the literature review in 2, an in-depth outline to our online parallel data extraction system is given. We will start by introducing the sentence representations used for extraction in 3.1, followed by the sentence scoring (3.2) and filtering techniques (3.3). We especially focus on finding an efficient, high accuracy extraction method without the introduction of any new hyperparameters. This exploration of different extraction methods is performed in the first set of experiments. After defining the data used for the experiments as well as their pre-processing (4.1), the first experiments, which are performed in an *unsupervised* setting without any parallel data, are presented (4.2). These do not only examine the performance of different extraction techniques, but also analyze the internal representations and their roles and impact on extraction as training progresses. Apart from a quantitative analysis, we will also show sample sentences extracted during training and provide a short qualitative analysis of the extraction. After having observed the unsupervised scenario, the best performing model will be used to explore the *semi-supervised* scenario where a small parallel base corpus is given for initialization 4.3. These will also incorporate a quantitative analysis of the representations and extracted pairs. Having explored our online-parallel sentence extraction in both unsupervised and semi-supervised settings, two smaller experiments focusing on different applications of the system, namely low-resource MT on Gujarati-English (4.4.1) and corpus filtering (4.4.2), will follow. At the end, a short summary of the major findings from the experiments will be given in addition to future research to undertake.

Chapter 2

Related Work

In this chapter, a brief insight into the history of MT, from rule-based MT (RBMT) over Statistical Machine Translation (SMT) to NMT is given. This is followed by a more concrete description of a popular NMT architecture. After then having mentioned several points of good-practice in NMT, covering topics such as the out-of-vocabulary (OOV) problem, batch creation and noise, a special focus will be placed on low-resource MT. As such, several approaches towards low-resource MT will be presented, ranging from pivot translation and multilingual NMT over parallel data extraction to the exploitation of monolingual data.

2.1 From Rule-Based to Statistical Machine Translation

The idea of using machines to translate between languages has existed for more than a century, and it was in the early 1930's that first patents on so-called **mechanical dictionaries** supplied first concrete concepts on this field (Hutchins, 2004). Georges Artsrouni developed a machine that would store source language words and their corresponding translations in several languages on a memory band, which could then be retrieved (Daumas, 1965). Almost simultaneously, Petr Trojanskij described a multilingual translation machine that would use abstract Esperanto-based symbols to encode grammatical functions between languages (Hutchins, 2006). This machine can be seen as a precursor of the idea of *interlinguality* in translation, which intends to add an abstract middle layer between two languages to be translated in order to make the translation process more generalizable between various language combinations. In fact, later we will see that the idea of interlinguality traverses the field of machine translation in the form of rule-based approaches based on abstract languages —so-called *interlinguas*— all the way to *interlingual representations* found in state-of-the-art NMT systems (see 2.2). More than a decade later, in 1949, Warren Weaver published his translation mem-

orandum (Weaver, 1949), in which he foresaw the potential of Claude Shannon's *Noisy Channel* model (Shannon, 1948) for automatic translation, which would open the doors of SMT and revolutionize the field at the end of the millennium.

In the 1950's and 60's, MT research bloomed and the major branches of **rule-based Machine Translation** came to be. The first one being the **direct translation** approach, consisting of rewriting rules to translate between a specific source and a target language using only a minimal amount of structural analysis. One early example is Gilbert King's method, which tries to solve lexical ambiguity simply by applying word-selection-rules based on the surrounding context words (King and Wieselman, 1956). Another approach is the **interlingua model**, which tries to overcome the problem of having to write new rules for every possible language pair by first analyzing the source language input and generalizing it to a language independent interlingua representation, which is then translated via rules into the target language. However, due to the complexity of the analysis needed to transfer a phrase into an interlingua and back, and the proneness to errors this yields, the idea of encoding into and decoding from an interlingua did not reach an effective implementation until the advent of modern multilingual NMT systems; in the form of *multilingual representations*. Lastly, **transfer models** can be thought of as the golden mean between the brute-force nature of direct translation models and the highly analytical interlingua models by first intending to disambiguate the source sentence –via a prior analysis taking into account its syntax and semantics— before transferring it into the target language (Hutchins, 2006, 376).

While RBMT yielded first practical implementations of automatic translation, they come attached with immense amounts of manual rule creation and updating. It was therefore a major change in practice when in 1990 a group of researchers at IBM published a first version of their **statistical machine translation** model, which would later be known as the IBM model (Brown et al., 1990). It essentially consists of a *language model*, using n-gram word probabilities, and a *translation model*. The latter of which uses, inter alia, alignment probabilities between source and target words. However, word-based models ignored the compositional nature of language and it thus did not take long until **phrase-based SMT** (PBSMT) systems were developed, outperforming word-based and even syntax-based systems (Koehn et al., 2003). In order to further generalize the behavior of semantically similar words as well as taking into account a larger context window than traditional discrete language models can, Bengio et al. (2003) developed a **neural network** based language model using distributed representations for words —*word-embeddings*— which was soon applied to SMT for re-ranking possible candidate translations (Schwenk et al., 2006). Similar approaches were used for re-ranking translations using neural-network-based translation models (Schwenk, 2012; Son et al., 2012) as well as joint models (Devlin et al., 2014).[1]

[1]The joint model uses a language model as well as a context window over the source sentence to calculate the probability of the next target word. This is quite similar to what happens in the attention mechanism of a modern sequence-to-sequence NMT system.

2.2 Neural Machine Translation

As neural network based components were introduced to SMT, the early 2010's also saw the emergence of first NMT systems consisting of jointly-trained neural networks capable of taking a source sentence as input and outputting a translated target sentence. A common architecture is the **encoder-decoder**, in which the former encodes a variable-length sequence into a fixed-length vector and the latter then decodes this vector back to a variable-length sequence. The first example of such a network introduced in Kalchbrenner and Blunsom (2013) uses a convolutional neural network (CNN) as the encoder, as well as an additional **recurrent neural network** (RNN) layer in their decoder in order to overcome the Markov Assumption that comes with conventional n-gram models. Although vanilla RNN's (Elman, 1991) are a good choice for modeling natural language sequences with large contexts, since they can "use their feedback connections to store representations of recent input events"(Hochreiter et al., 2001, 2), they do come with major practical flaws such as the *vanishing gradient* problem[2] which makes training basic RNN's and maintaining long-range dependencies difficult. In a similar encoder-decoder framework, Sutskever et al. (2014) used **long short-term-memory** (LSTM) (Hochreiter and Schmidhuber, 1997) units to surmount the limitations of simple RNNs and thus to further improve the handling of long-range dependencies. Their sequence-to-sequence model is also sensible to word-order, as opposed to Kalchbrenner and Blunsom (2013)'s convolutional encoder. Alternatively to LSTMs, **gated recurrent units** (GRU) have also been used for these purposes (Cho et al., 2014b), although recent findings suggest that GRUs tend to be outperformed by LSTMs for this task (Britz et al., 2017).

2.2.1 Model Description

The encoder-decoder framework common to Bahdanau et al. (2014), Cho et al. (2014b), Sutskever et al. (2014) and many others relies on the following principles.[3]

The encoder reads the source sequence $\mathbf{x} = (x_1, ..., x_{T_x}), x_i \in \mathbb{R}^{K_x}$, where T_x is the total number of *time steps* —where each token in a sentence constitutes a time step— in the current sequence and K_x is the vocabulary size of the source language, into a vector $\mathbf{c}$. In a first step, a continuous representation v_{x_i} of element x_i —which in this description is a one-hot word vector but could also be a word index— is retrieved from a **word embedding matrix** $\bar{E} \in \mathbb{R}^{m \times K_x}$, where m is the word embedding dimensionality, as such:

[2]The vanishing gradient problem is such that "the influence of a given input on the hidden layer, and therefore on the network output, either decays or blows up exponentially as it cycles around the network's recurrent connections." (Graves, 2012, 31)

[3]The notation in this section is based on Bahdanau et al. (2014) and Graves (2013).

$$v_{x_i} = \bar{E}x_i \tag{2.1}$$

The new sequence of continuous representations $\mathbf{v} = (v_{x_i}, ...v_{x_{T_x}})$ is then **encoded** into a fixed-length vector. For this purpose, an RNN is a common choice, such that the output vector is the last **hidden state** h_{T_x}, which acts as a summary of the input sentence. An RNN's hidden states are defined as:

$$h_t = f(v_{x_t}, h_{t-1}) \tag{2.2}$$

where $h_t \in \mathbb{R}$ is the RNN's hidden state at time step t and f is a non-linear function such as a GRU (Cho et al., 2014b), LSTM (Sutskever et al., 2014) or even a simple sigmoid or tanh (Kalchbrenner and Blunsom, 2013). **LSTM**s are used for our experiments, which, as described by Graves (2013), calculate h_t as:

$$h_t = o_t \tanh(c_t) \tag{2.3}$$

where o_t is the *output gate* and c_t the *cell state*, defined as:

$$o_t = \sigma(W_{vo}v_{x_t} + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{2.4}$$
$$c_t = f_t c_{t-1} + i_t \tanh(W_{vc}v_{x_t} + W_{hc}h_{t-1} + b_c) \tag{2.5}$$

where σ is the logistic sigmoid function, the various Ws are weight matrices[4], b_o and b_c the bias terms, f_t the *forget gate* and i_t the *input gate*, as such:

$$f_t = \sigma(W_{vf}v_{x_t} + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{2.6}$$
$$i_t = \sigma(W_{vi}v_{x_t} + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{2.7}$$

where, again, the Ws and b_f, b_i are the weights and biases respectively. Once all elements of the input sequence have been fed through the encoder, the last hidden state h_{T_x} is returned.

The task of the **decoder** is then to iteratively predict the target words y_t given the last hidden state h_{T_x}, the current decoder state s_t and the previously generated word

[4]The subscripts of the weight matrices, for example W_{vo}, indicate the corresponding vector (v_{x_t} in this example) and the current gate or cell state being calculated (o_t etc.).

y_{t-1}. The probability $p(y_t)$ of all possible target words at time step t is then modeled as such:

$$p(y_t) = g(y_{t-1}, s_t, h_{T_x}) \tag{2.8}$$

where g can, similarly to the encoder, be an LSTM, GRU or other nonlinear function that returns the vector of probabilities for the current time step. The softmax function can then be used over all entries in $p(y_t)$ to find the most probable target word.

Thus, the decoder assigns probabilities to potential target word candidates based on the source sequence and previously generated words. In this setup, it takes the information about the source sentence from the last hidden state of the encoder only. However, reading all of the target sentence from a single fixed-length vector leads to a degrading translation quality as source sentences become more complex (Cho et al., 2014a). To overcome this problem, **attention mechanisms**, which allow the decoder to attend on various parts of the source sentence before generating a target word, came to be.

The two major types of attention mechanisms are the *multiplicative* (Luong et al., 2015a) and *additive* (Bahdanau et al., 2014) approaches. While the former allows the model to attend on a preselected window over the source hidden states, the latter, while being more computationally expensive, allows the model to attend freely over the hidden states of the complete source sequence, which leads to a slightly better performance (Britz et al., 2017). In order to perform attention, the encoder returns a set $\mathbf{h}$ containing each hidden state at each time step, such that:

$$\mathbf{h} = \{h_1, ..., h_{T_x}\} \tag{2.9}$$

In Bahdanau et al. (2014), a **bidirectional RNN** (BRNN) is used, which consists of a forward and a backward RNN. While the forward RNN reads in the sequence from x_1 to x_{T_x} returning forward hidden states $(\overrightarrow{h}_1, ..., \overrightarrow{h}_{T_x})$, the backward RNN reads the sequence in the opposite direction returning backward hidden states $(\overleftarrow{h}_1, ..., \overleftarrow{h}_{T_x})$. In such a setting, the hidden states passed to $\mathbf{h}$ are defined as the concatenation of the forward and backward hidden state for each source word x_i, such that:

$$h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i] \tag{2.10}$$

Since each element's hidden state is directly influenced by the time steps preceding (and in the case of the BRNN also the steps following it), they carry contextual

information for every word and can therefore be referred to as *context vectors*.[5] In a global attention mechanism, these context vectors are assigned different weights by an *alignment model* based on the previous decoder hidden state. Instead of the last hidden state only, these newly weighted encoder hidden states are then passed to the decoder at each time step i:

$$p(y_i) = g(y_{i-1}, s_i, c_i) \tag{2.11}$$

where y_{i-1} is the previously generated word, s_i the current decoder state, and c_i the *weighted encoder hidden states*, defined by Bahdanau et al. (2014) as:

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j \tag{2.12}$$

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{2.13}$$

$$e_{ij} = a(s_{i-1}, h_j) \tag{2.14}$$

Where a is a feed forward neural network. As such, the decoder does not rely on the lossy sentence representation of the last encoder hidden state, but can instead search through each hidden state corresponding to a source word and its context before generating a target word y_t.

As attention mechanism brought a major boost in performance of RNN-based NMT systems, an alternative model came about that does away with RNNs completely by replacing them with several stacks of *self-attention* over the source and target sequence; the so-called **transformer**, whose architecture is best described in the original paper by Vaswani et al. (2017). Additionally to the LSTM-based models, we will also apply this newer architecture in our experiments.

2.3 Good Practices in NMT

As is common to most machine learning approaches, good practices to overcome limitations and improve overall performance of a specific task are explored by the research

[5]Note that the term *context vector* is an ambiguous term in the field of NMT, as it has been used to describe various components of sequence-to-sequence models such as the output of the attention mechanism in (Bahdanau et al., 2014) or as the set of encoder outputs (España-Bonet et al., 2017). For our purposes the latter definition is intended.

community over time. In this section, some of the major points towards a good practice in NMT are described.

2.3.1 The Out of Vocabulary Problem

Natural languages come with large vocabularies, with modern dictionaries often citing more than hundreds of thousands of words. Taking into account that in standard MT, every word form —or *type*— has its own vocabulary entry, these dictionary sizes quickly surpass the possibilities of computation —especially for languages with complex morphological forms such as inflection, compounding or affixation— as NMT vocabulary sizes are usually limited within the ten thousands. A simple way to overcome this problem is to define a limited vocabulary and replace all out-of-vocabulary (OOV) words with a reserved symbol. Nevertheless, the translation quality suffers the larger the number of unknown words (Cho et al., 2014a). It has therefore been the object of MT research to tackle the OOV problem with more elaborate techniques.

Such techniques include —but are not limited to— performing softmax over a varying subset of a large vocabulary during training and decoding (Jean et al., 2015), or copying unknown source words directly into the target sentence or using back-off dictionaries (Luong et al., 2015b). A simple but efficient technique is using **subword units** as part of the vocabulary, so that morphologically complex structures and named entities can be decomposed into simpler units, thus decreasing the vocabulary size immensely. The context, that the source subword units are embedded in, is then exploited by the NMT to decode them into corresponding target units. For the purpose of finding appropriate subword units, unsupervised morpheme induction has been suggested (Virpioja et al., 2007; Stallard et al., 2012), while Sennrich et al. (2016b) adapt *byte-pair encoding* (**BPE**) (Gage, 1994), a data compression algorithm that iteratively merges the most frequent adjacently occurring bytes in a sequence with a single, unused byte. Similarly, when BPE is applied to characters instead of bytes, the result is a mix of frequent words and subword units. While BPE is not linguistically motivated, it has the major advantage of being open-vocabulary, as any previously unseen data can undergo the same merge operations as the training data so that no unseen units are encountered during testing.

Due to its simple yet efficient solution of the out-of-vocabulary problem, BPE is also used in our experiments (see 4.1).

2.3.2 Batch Creation

For increased efficiency and quality, NMT models do not backpropagate after each source-target sentence pair, but instead train on mini-batches —often simply called **batches**— of sentences, updating parameters between each batch. Within each batch,

sentences of varying lengths might be included, where shorter sentences need to be **padded** with zero-entries to match the length of the longest sequence within the same batch. As computational cost is thus proportional to the longest sentence in a batch, reducing the number of long-sequence batches by creating batches holding similar length sequences is a common practice to speedup the training process (Sutskever et al., 2014).[6] While there are no major differences in accuracy, unsorted batches or source sentence length ordered batches seem to have faster dropping perplexities than other sorting methods when using the optimization algorithm *Adam* (Kingma and Ba, 2015). Using *stochastic gradient descent* (SGD), the sorting method has little effect on the perplexity drop, thus using the most efficient sorting method, namely sorting on the target sentence length and breaking ties via the source sentence length, is a good option (Morishita et al., 2017).

While such sorting can be performed on the corpus directly before dividing it into batches, this leads to the same sentences appearing next to each other during training as well as favors similar sentences to appear in the same batch, as utterances of specific sizes tend to share features (Doetsch et al., 2017). **Bucketing**, a method where sentences of similar length are placed in the same bucket, so that at each epoch the model can draw random samples from a bucket to create a batch of varying sentences with similar length, is a popular method to overcome the above limitation.

As for the number of sentences per batch, larger batches[7] seem to lead to more stable gradients as more sentences have an impact on the update, having a positive impact on accuracy (Morishita et al., 2017). The same positive impact of larger batch sizes has also been observed for Transformer models (Popel and Bojar, 2018).

For our purposes, drawing batches of 64 (LSTM) or 50 (Transformer) sentences from source length sorted buckets was chosen.

2.3.3 Model Hyperparameters

When training neural networks, one is confronted with the task of finding appropriate hyperparameter settings, usually led by a mixture of learning from previous work, tuning and intuition. As for NMT, some investigation has been done in hope to create a canon of good practice for setting hyperparameters.

Embedding dimensionality as one hyperparameter of a RNN-based NMT system tend to perform slightly better with increasing size. Nevertheless, comparing a 2048

[6] Sutskever et al. (2014) observed that a model with sentence-length ordered batches finished computation in half the time as opposed to a model with randomly ordered batches.

[7] As 64 is the largest batch size investigated by Morishita et al. (2017) due to memory constraints, it is not clear if even larger batches have the same effect. One major concern is that with even larger batches, the number of parameter updates in a time frame decreases, slowing convergence.

dimensional embedding to one of 128 dimensions, Britz et al. (2017) report that the larger embedding outperformed the smaller only by a small margin[8] while taking twice the time to reach convergence.

Another variable of an RNN-based NMT model is the **depth** of its encoder and decoder. The number of hidden layers of successful NMT models usually ranges from 1 (Grégoire and Langlais, 2018) up to 8 (Luong et al., 2015b). Nevertheless, while there seems to be a tendency to assume that deeper RNNs outperform shallow ones (Sutskever et al., 2014), the experiments of Britz et al. (2017) suggest that this does not hold true for the encoder, since no statistically significant boost in performance was observed when increasing the encoder depth to a number larger than 2. They further showed that BRNNs do not significantly outperform unidirectional RNNs.

For Transformer models, larger models —e.g. with an increased size of the feed-forward layers and a larger number of heads— generally outperform smaller models. However, these take longer to converge and may lead to out-of-memory errors. In that case, using a smaller model with a larger batch size is preferred (Popel and Bojar, 2018).

Regarding the **maximum sequence length** passed to a Transformer model, Popel and Bojar (2018) find that setting this value too low[9] will result in a decay in translation quality, as a larger number of the training corpus may be excluded and the model will not be able to decode sentences longer than the maximum length set during training, degrading its performance at test time. Combining a larger batch size of 2000 tokens with a reasonably large maximum length of 70 is therefore advisable.

As for the **learning rate**, Popel and Bojar (2018) further show that a learning rate of $0.05 - 0.25$ yields similarly good results. In case the learning rate is set too high, the models optimizer may jump over local minima and wander off into a plateau, leading to a failed training. In such cases, decreasing the learning rate as well as applying *gradient clipping* is advisable next to increasing the *warm-up steps* of the `noam` schedule used in Transformer models.

Lastly, Popel and Bojar (2018) also show that training a transformer on a larger amount of Graphics Processing Unit (**GPU**) does not only reduce computational speed immensely, but it also increases the translation performance.

2.3.4 Noise

When training NMT models with noisy data, such as for example comparable corpora based on crawled data from the web, the models performance can be weakened.

[8] They report an asymptotic significance of $p = 0.01$ only.

[9] Where *low* stands for values around >= 70 for models using smaller batch sizes of 1500 tokens and >= 50 for those using a larger batch size of 2000 tokens.

However, the impact of noise on NMT depends on the kind of noise. It seems that NMT is quite robust against very short sequences or seeing the wrong source language, while training on untranslated target sentences has a high impact (Khayrallah and Koehn, 2018) as it quickly turns the NMT system into an autoencoder.[10] A less obvious noise in crawled data is machine translated content itself, as it often includes unnatural phrasing and is prone to errors (Antonova and Misyurev, 2011).[11]

As we envision that our extraction model can identify useful sentences, we did not apply any language detection as pre-processing to filter out untranslated sentences. The results of our model applied to filtering can be seen in 4.4.2

2.4 Multilingual Machine Translation

The idea of Multilingual MT is based on single models that strive to translate between several languages, instead of between only one source and target pair. They are related to *multitask learning* (Johnson et al., 2017) —or further, multitask representation learning— as they learn generalized internal representations that cover various *tasks* (here, translating between various languages) via a shared common ground (here, the underlying semantics of sentences). It has been shown that this type of learning is especially beneficial for low-resource scenarios, as the general representations learned on other, yet very similar, tasks helps the model perform significantly better on a task with little or even no data available (Maurer et al., 2016).

Until recently, multilingual MT systems often came with language specific encoders or decoders. In Zoph and Knight (2016), a **many-to-one** system is proposed, which uses a separate encoder for each source language. In order to combine the hidden states of the otherwise separated encoders, a combination layer is added on top. Thus, in order to train this system, a multilingual parallel corpus is required.[12] The **many-to-many** approach by Firat et al. (2016a) overcomes the necessity of a multi-way parallel corpus by introducing an attention mechanism over its encoders, removing the constraint of having to train each encoder simultaneously. Their multilingual model outperforms comparable one-way MT models, especially for low-resource language pairs. Nevertheless —similarly to Zoph and Knight (2016) and the many-to-one model by Dong et al. (2015)— separate encoders and/or decoders are needed to perform multilingual translation, making the model architecture large and computation much more complex.

[10] Khayrallah and Koehn (2018) note that with 5% untranslated target sentence noise, the performance drops by 9.6 BLEU and with 20% of the same noise during training, the number of copied sentences during testing is catapulted to about 60%.

[11] Rarrick et al. (2011) showed that the amount of MT content on the web varies by language, with high-density languages such as German having only a "small percentage" of MT content versus low-density languages such as Lithuanian or Latvian having an MT content of about 50%.

[12] That is, for a 2-1 translation setting, a 3-way parallel corpus is needed.

The solution to the large number of components of multilingual MT systems proved to be simple. Instead of using separate encoders and decoders for each language, **language tokens** specifying the corresponding target language are added to the beginning of each source language.[13] In such a setting, a simple MT system with one encoder and decoder suffices, as the model learns to translate to the corresponding target language by taking into account the language token in the source sequence (Ha et al., 2016; Johnson et al., 2017).

When looking at the attention-weighted encoder states at each decoding step, it can be observed that source sentences and their translations appear closely in the embedding space. As such, multilingual models trained in a single encoder-decoder framework can yield **interlingual representations**. In the same vein as multitask representation learning, these have been proven to be useful for performing translation between languages that have no parallel data, so-called **zero-shot MT**. Concretely speaking, this is due to the fact that, even though two languages share no parallel data, training them in the same system with existing parallel data to other *pivot* languages yields reasonable interlingual representations for these languages, enforcing that semantically similar sentences of the two languages without parallel data appear in similar regions of the embedding space. Because similar words are encoded closely to each other, the model can even decode sentences which are partially encoded in different languages, thus being robust against *code switching* (Johnson et al., 2017).

In section 3, an outline of how to use these interlingual representations to extract parallel data from a comparable corpus will be given.

2.5 Low-Resource Machine Translation

In the above section we have seen that multilingual MT, when the internal representations are shared, is beneficial for low-resource MT. However, this is only one approach to tackling the scarce availability of parallel data.

2.5.1 Pivot Languages and Multilingual NMT

Early approaches towards low-resource MT included using so-called **pivot languages**, languages for which rich parallel data is available to both sides of a low-resource language pair. In SMT, a model for a low-resource pair could be estimated from the translation probabilities of the source-pivot and pivot-target models (Wu and Wang, 2007). Alternatively, sentences can be simply translated from the source to a pivot language and from

[13] Another token specifying the source language can also be added. Nevertheless, MT models seem to learn to recognize the source language and treat it appropriately if the source token is missing.

there to the target using two separate SMT systems (Utiyama and Isahara, 2007). In both cases, the pivot language is inserted as an intermediate step within the translation path of the low-resource source and target pair. Further, generated translations from the source to the pivot can be used to train a many-to-one multilingual NMT system similar to Zoph and Knight (2016) on a pseudo multi-way parallel corpus from the low-resource source and pivot to the target, in order to initialize and train a *zero-resource* source and target pair (Firat et al., 2016b).

However, pivot languages can also be thought of as appearing in **multilingual MT** models (see 2.4) in the form of high-resource languages that enrich the interlingual representations. Nevertheless, multilingual NMT systems tend to decrease in performance as the number of covered languages increases due to the growing number of languages that need to be represented in a finite vocabulary size (Lakew et al., 2018). Zoph et al. (2016) suggest a transfer learning approach in which a small parallel corpus of a low-resource language pair is used for training on top of a trained high-resource model. In such a setting, the model is trained to maximize its performance on the low-resource pair only, yielding better results on its designated low-resource pair than a multilingual model trained to maximize accuracy on all its covered languages.

2.5.2 Parallel Data Extraction

As seen in 2.5.1, multilingual NMT can improve the performance of low-resource language pairs. However, these still perform poor in comparison to their high-resourced language counterparts. As such, increasing the amount of available parallel data has become a major objective of low-resource NMT research. For these purposes, a **comparable corpus**, which is usually a collection of monolingual source and target sentences in similar domains, is needed. Such a comparable corpus often consists of a large amount of articles of similar content in the languages of interest, as these can systematically be crawled from the web. The following task is to identify and extract potentially parallel sentences from the comparable corpus.

Early approaches for the detection of parallel sentences in non-parallel corpora were mostly **statistical** models, including the use of maximum entropy classifiers (Munteanu and Marcu, 2005), measures based on cross-language information retrieval (Utiyama and Isahara, 2003), or conditional random fields (Smith et al., 2010). Or **structural** approaches with word-to-word lexicon look-ups or the exploitation of the structure of general web-pages (Resnik and Smith, 2003) or Wikipedia[14](Adafre and de Rijke, 2006).

However, most of the above-mentioned methods rely on either feature engineering, which may be language or resource dependent, or the prior analysis of the document structure. Following the idea that a pre-trained SMT system already includes a good

[14]`https://www.wikipedia.org/`

dictionary —in the form of translation probabilities over words— and that language-independent metrics —such as the *word error rate*[15] or *translation edit rate* (Snover et al., 2006)— can easily be applied to their outputs, Abdul-Rauf and Schwenk (2009) suggest to use a **SMT** system for parallel sentence extraction. For these purposes, the source sentences of a comparable corpus are translated to the target language using a pre-trained SMT system. Using these generated target sentences to perform information retrieval over the actual target side of the comparable corpus, target candidates are found, from which final source-target pairs are then extracted taking into account different features such as the word error rate or length discrepancies. This approach can be performed iteratively. That is, after the pre-trained SMT system has been used to extract new parallel data from the comparable corpus, it can be trained on these sentences, which will further enhance the extraction accuracy in a later iteration. It can therefore be considered a type of *online* extraction, much as we envision.

While Snover et al. (2006) exploit SMT translations to find additional parallel data, España-Bonet and Barrón-Cedeño (2017) as well as España-Bonet et al. (2017) see potential within **interlingual representations** found in multilingual NMT systems for the task of parallel sentence extraction. Based on that idea, Grégoire and Langlais (2018) train a siamese network (Bromley et al., 1993) on a *pseudo-parallel corpus*[16], such that its two encoders read in the source and target sequences. Their last hidden states are then sent through a *feed-forward neural network*, which is trained to classify a pair of hidden states as either being parallel or not. In later experiments, they use this trained classifier to extract parallel sentences from an actual comparable corpus.

Similarly, Schwenk (2018) use the encoder of a pre-trained NMT system to embed sentences into multilingual space, taking a threshold over the cosine similarity over these sentence representations to extract parallel sentences, while Bouamor and Sajjad (2018) use cosine similarity over the averaged word embeddings of each sentence to filter a list of parallel sentence candidates.

Artetxe and Schwenk (2018) observe that cosine similarity of sentence representations tend to be scaled differently, leading to some sentences without any true translations in the corpus to have a generally high cosine similarity to many sentences, while some sentences with a true translation have a much lower cosine similarity to their corresponding translation. As such, they explored different **margin-based scoring** techniques that would overcome the scaling problem of cosine similarity. (More in 3.2.)

The approach outlined by Schwenk (2018) or Artetxe and Schwenk (2018) are not intended to be iterative, as the classifier is only trained once on parallel data before the decoder is discarded and their extraction procedure commences. The idea of using

[15]The word error rate is a measure of how similar two strings are and takes into account any substitutions, insertions or deletions between a tested string and its reference.

[16]A pseudo-parallel corpus is a corpus that consists of parallel sentences which have been shuffled among each other —potentially with additional negative samples— to create non-parallel sentences.

interlingual representations found within an NMT system, however, has the potential to be useful in an **iterative setup**. When an NMT model is trained on a small amount of parallel data and its resulting representations over a comparable corpus are used to detect parallel sentences, training the system on these will potentially improve the system's performance and quality of the interlingual embeddings, leading to more accurate parallel sentence extraction in later iterations. This process is the objective of our research and shall be described in detail in section 3.

2.5.3 Exploiting Monolingual Data

Extracting parallel data from comparable corpora is one approach to using non-parallel sources in order to improve low-resource NMT. An alternative approach that does not rely on extraction decisions, is to train directly on ample amounts of monolingual data.

In order to include a **language model** (LM) trained on monolingual data only, Gulcehre et al. (2015) suggest to concatenate the NMT decoder hidden states with those of an RNN-based LM. While this improves the performance of both low and high resource translation, the architecture of the fused NMT architecture and LM is complex.

NMT systems already condition the generation of a target word on its predecessors, which should make the use of an additional LM avoidable. As such, one simple yet effective technique to incorporate monolingual data during training, is to use **back-translation**. A monolingual target language corpus can be machine translated into the source language to create additional parallel data to train on. As the target language is still an original sentence from the monolingual corpus, the systems fluency and overall performance is improved (Sennrich et al., 2016a). However, the quality of the pseudo-parallel training data generated in such a setup depends largely on the existence of a decently trained auxiliary MT (possible SMT) system. Applying a method that is independent of such auxiliary models, Currey et al. (2017) show that adding a simple target sentence copying task also significantly improves fluency.

In order to include quality control when training on machine translated monolingual data, He et al. (2016) created a reinforcement learning-based **auto-encoding** task, where one MT model translates a monolingual sentence into the target, which is then checked for fluency in the target language, before translating it back to the source language using a different MT model. At the end, they check whether the original source and the generated source are consistent and feedback is given to the two MT models. While their setup improves significantly over the back-translation approach by Sennrich et al. (2016a), its smallest tested model still required hundreds of thousand of parallel sentences.

Having no parallel data at hand, Irvine and Callison-Burch (2016) combine existent

bilingual dictionaries and **dictionary induction** techniques on monolingual data to create a word-by-word translation model and to improve the vocabulary coverage of a low-resource SMT model. Initializing their system on a similar word-by-word translation model (Lample et al., 2018b) induced on monolingual data, Lample et al. (2018a) train increasingly complex NMT models on monolingual data only by training them on a **denoising task** as well as back-translated data of earlier models. Additionally, they enforce the encoder to project the sentences in an interlingual space using a generative adversarial network (GAN) structure, in which the model is rewarded when it manages to trick a discriminator that tries to predict the encoded language. With this setting, the unsupervised model learns to translate between a zero-shot language pair with suprisingly good accuracy.[17] In a similar setup, Artetxe et al. (2018b) initialize their NMT system with multilingual embeddings —learned from monolingual data and projected in a common space— before using denoising and backtranslation to train an unsupervised model.

Combining the idea of unsupervised multilingual embeddings with the observation that SMT systems generally outperform NMT systems when data is sparse, Artetxe et al. (2018a) train cross-lingual embeddings to induce a phrase table. In order to use this phrase table in a standard **PBSMT** setup, it is combined with a LM and a distortion model. The model's weights are then iteratively tuned through a combination of *minimum error rate training* and backtranslation, yielding a purely unsupervised approach. In a similar fashion, Lample et al. (2018c) apply embeddings-based initialization, language modeling and back translation for both PBSMT and NMT, with their combined PBSMT-NMT model being the current state-of-the-art in unsupervised MT.

2.6 Domain Adaptation

A very related field to low-resource MT is domain adaptation. As general parallel corpora are already sparse for most language combinations, the availability of adequate in-domain data is especially meager. In order to adapt MT models towards the domain of interest despite the lack of in-domain data, several approaches emerged. Applying **backtranslation**, an existing MT model trained on a general corpus, can be used to translate monolingual in-domain corpora, which can then be used as additional training data (Schwenk, 2008; Sennrich et al., 2016a). In case the language pair is generally low-resourced, such that general MT models do not produce sufficient quality translations, human **post-editing** is required (e.g. Nisarg et al. (2018)), which quickly becomes costly.

[17] They report that "on the WMT dataset we [they] can achieve the same translation quality of a similar [neural] machine translation system trained with full supervision on 100,000 sentence pairs." (Lample et al., 2018a)

An alternative approach is to select in-domain sentences from large general corpora using statistical methods, such as **extracting sentences** from a general corpus which have a low cross-entropy with an in-domain LM (Axelrod et al., 2011). Similarly, in-domain sentence pairs can also be extracted from comparable corpora. For this, Rauf and Schwenk (2011) use an IR-based approach where back-translated sentences are used as queries over a comparable corpus to obtain similar sentences, while Barrón-Cedeño et al. (2015) use a variety of text similarity measures.

Chapter 3

Online Parallel Data Extraction

In the previous section we have introduced various methods towards parallel sentence extraction. In España-Bonet et al. (2017); Schwenk (2018); Artetxe and Schwenk (2018) etc., it has been shown that similarity measures over interlingual sentence representations that can be found within sequence-to-sequence NMT systems can be used to identify parallel sentences in comparable corpora.

Yet, all of the above systems dispose of their classifiers once enough data has been collected to train a separate NMT system. However, España-Bonet et al. (2017) has empirically shown that sentences which are translations of one another tend to move towards each other as training progresses. As representations of translations move closer to each other in the multilingual space, the confidence of them being parallel grows, yielding a better classifier.

The premise of **online parallel data extraction** is therefore that NMT systems — either sequence to sequence models with RNNs, transformers, or any architecture based on encoder-decoder models— already learn strong enough representations of words and sentences to judge on-line if an input sentence pair is useful or not. Starting with an NMT system that has been initialized —either via pre-trained word embeddings only or by pre-training the model on a small parallel corpus— first sentences are extracted from a non-parallel corpus. Whenever enough sentences have been selected to create a batch, a train step is performed. Embeddings are modified by back-propagation. This again improves both the translation quality and the internal representations, which also leads to a better classifier which can find more parallel sentences with higher confidence. As such, as we train and extract in a loop, both tasks enhance each other, leading to better extraction and translation quality. Notice that the extracted pairs differ through iterations, since it is the sentence representation at the specific training step that is responsible for the selection.

This approach also resembles **self-supervised learning** (Raina et al., 2007; Bengio et al., 2013), i.e. learning a *primary task* (PT) where labeled data is not directly available but where the data itself provides the supervisory signal for another *auxiliary task* (AT) which lets the network learn the PT. Often, the PT involves representation learning, as is the case with the multilingual representations found in NMT systems. In the case of our online-extraction approach, the learning approach comes with a twist: Extracting cross-lingually close sentences comes as an AT for learning MT and learning MT comes as an AT for finding cross-lingually close sentences in a mutually self-supervised loop; in effect a *doubly virtuous circle*.

3.1 Sentence Representations

Semantics of words and sentences can be represented in a variety of ways. In our case, we rely on so-called **distributed semantics**. It relies on the idea that words that occur in similar contexts tend to have similar meanings. Or as Frege proclaims more radically:

"Nur im Zusammenhange eines Satzes bedeuten die Wörter etwas. [Only in the context of a sentence do words have meaning.]" (Frege, 1884)[§62]

The sentence representations we focus on can also be regarded as being **compositional**, as they are based on underlying word representations which were composed in a certain way to yield the sentence representation. However, for our purposes there are several composition methods available used to obtain sentence representations, ranging from taking the sum or the mean (España-Bonet et al., 2017) or performing max-pooling (Schwenk, 2018) over encoder outputs, over taking the last hidden state of a BRNN (Grégoire and Langlais, 2018) to calculating the average of the word embeddings (Bouamor and Sajjad, 2018).

Our experiments are based on the composing method defined by España-Bonet et al. (2017) due to its strong empirical foundation. Instead of focusing on solely one representation type found in encoders, a focus is taken on two different embedding spaces to build semantic sentence representations. Namely, the sum of the word embeddings (C_e) and the hidden states of an RNN or the encoder outputs of a transformer (C_h):

$$C_e = \sum_{t=1}^{T} e_t \tag{3.1}$$

$$C_h = \sum_{t=1}^{T} h_t \tag{3.2}$$

where e_t is the word embedding at time step t and h_t its hidden state (RNN) or

encoder output (transformer). In case h_t is an RNN hidden state, it is further defined by the concatenation of its forward and backward component $h_t^{\texttt{RNN}} = [\overrightarrow{h}_t; \overleftarrow{h}_t]$

Both representation types are of interest, since they have different **modeling capabilities**. As C_h is based on the processed sequence of the encoder, where each component at a time-step is influenced by its surrounding time-steps, it is capable of modeling these rather *fluid* dependencies. Also, as they are usually randomly initialized at the beginning of training, their values usually do not yet contain extreme-values. That makes them easier to adapt to the training data and overall gives them a more *flexible* nature (see 4.2.2.1). On the other hand, each word-embedding used to compose C_e is not context-dependent, which leads to a more *discrete* representation in C_e. Additionally, when C_e is already pre-trained, it contains more larger-valued entries, which makes them slower to adapt via gradient-descent, which results in C_e being more *rigid* over the epochs. Lastly, when C_e is pre-trained, it finds much more parallel candidates at the beginning of training compared to C_h, making it more *permissive* at that point. (See 4.2.2.1 for a detailed discussions)

3.2 Scoring

In order to find translations among all the possible $n \times m$ combinations of source-target sentence pairs, a scoring function is needed. We explore two such functions in experiments I.

Let $S_{\texttt{L1}}$ and $S_{\texttt{L2}}$ be the vector representations for each sentence of a pair (either C_e or C_h). The **cosine similarity** of a sentence pair is then calculated as:

$$\text{sim}(S_{\text{L1}}, S_{\text{L2}}) = \frac{S_{\text{L1}} \cdot S_{\text{L2}}}{||S_{\text{L1}}||||S_{\text{L2}}||} \tag{3.3}$$

After calculating the cosine similarity over all possible source-target combinations, a hard-threshold over the scores could be used to determine whether a sentence pair should be accepted or not. However, this threshold is not straightforward and depends on the language pair and the corpus (España-Bonet et al., 2017). Additionally, the measure might **scale differently** for different sentences (Artetxe and Schwenk, 2018). This leads to some sentences having a high cosine similarity with many non-translations, while true-translations might have a comparatively low cosine-similarity with one sentence. Three possible reasons that come to mind are:

1. Especially at the beginning of training, the semantic representations are often **sparse**. Even if word-embeddings are pre-trained, some words in the corpus used

to train the embeddings might not have been frequently used in the same context as in the corpus used for training the NMT system. If a word did not appear in the embedding training corpus at all, its embedding is randomly initialized close to 0. When now taking the cosine similarity between such sparse representations, the cosine similarity will be high to many other randomly initialized embeddings, making them difficult to keep apart.

2. The above is enforced by the problem that word-embeddings commonly used within NMT systems (`word2vec` etc.) cannot model **polysemy**. When now taking the cosine similarity between such unadapted representations, it can lead to high-values where they would usually not be expected by a human observer.

3. Word-embeddings usually do not have zero-mean and are **anisotropic**. That is, the variance of the representations is not the same over all dimensions. In fact, often most of the variance is accounted for in a low dimensional subspace of the representations, making the values in all other dimensions very close to each other (Mu and Viswanath, 2018). Initializing the NMT model with pre-processed word embeddings that are isotropic with zero-mean might help alleviate the problem of scalability when using cosine similarity.

To overcome the problem of scalability, Artetxe and Schwenk (2018), proposed a margin-based function:

$$\text{margin}(S_{\text{L1}}, S_{\text{L2}}) = \frac{\text{sim}(S_{\text{L1}}, S_{\text{L2}})}{\text{avg}_{\text{NN}_k}(S_{\text{L1}}, P_k)/2 + \text{avg}_{\text{NN}_k}(S_{\text{L2}}, Q_k)/2} \tag{3.4}$$

where $\text{avg}_{\text{NN}_k}(X, Y_k)$ corresponds to the average similarity between a sentence X and $\text{NN}_k(X)$, i.e. its k nearest neighbors Y_k in the other language:

$$\text{avg}_{\text{NN}_k}(X, Y_k) = \sum_{Y \in \text{NN}_k(X)} \frac{\text{sim}(X, Y)}{k} \tag{3.5}$$

This scoring method penalizes sentences which have a generally high cosine similarity with several candidates. Consequently, it highlights sentences which have a relatively high cosine similarity with one candidate, while maintaining low scores with all others. Following Artetxe and Schwenk (2018), all our experiments using the margin-based scoring function use $k = 4$.

However, since this function takes into account the k-nearest neighbors of each sentence, small input documents lead to scarce information of a sentence's relation to candidate translations. This makes the function less reliable for very small documents,

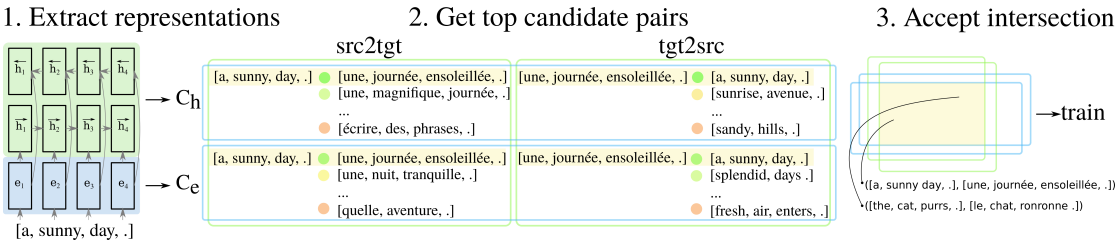


Figure 3.1: Main sentence selection method. First, two sentence representations of a sentence, C_h and C_e, are extracted. For each representation type, the top-scoring target sentence for each source sentence and vice versa are found. Only pairs that have been ranked as top-candidate pairs in both directions and both representation types are accepted as parallel pairs and collected for training. Whenever enough parallel sentences are extracted to create a batch, a train step is performed.

potentially accepting sentences with poor representations as good candidates simply due to this statistical uncertainty. Merging small documents into lots of at least 15 sentences per language is enough to solve this problem.

3.3 Filtering

After having scored all possible source-target combinations, it is important to filter for possible translations. In the following, four different filtering methods are introduced. In all of them, $\text{sim}(S_{\text{L1}}, S_{\text{L2}})$ and $\text{margin}(S_{\text{L1}}, S_{\text{L2}})$ can be used for scoring.

1. **Threshold Dependent.** The highest scoring target for each source sentence (pair i) as well as the highest scoring source for each target (pair j) for either representations $S = C_h$ *or* $S = C_e$ (systems H and E respectively in the following experiments) are identified. Since in general it is possible that the top target for a source sentence does not have the source as its top candidate and vice versa ($i \neq j$), this process is not symmetric. Only pairs that have been matched during selection in *both* language directions are accepted to the candidate list.[1] This first intersection is the **primary filter**, as it performs a first selection of source-target pairs, removing the majority of unwanted combinations.

   After having matched pairs, a threshold can be used to filter out additional false positives. It can therefore be seen as the **secondary filter** for the *single representation* systems H and E. However, such a threshold is an additional hyperparameter to tune and varies strongly with the corpus, model architecture and initialization techniques used.

[1] This differs from the approach in Artetxe and Schwenk (2018) in the fact that they use the *union* between pairs i and j, while here the *intersection* is used.

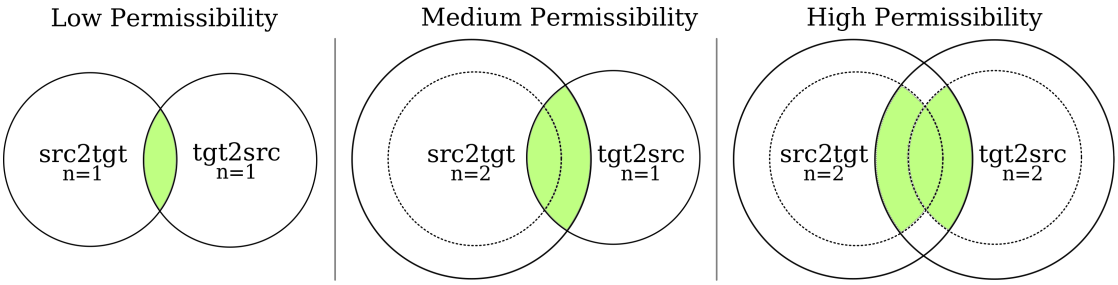


Figure 3.2: The system with low permissibility (left) accepts sentences which are top-candidates in *both* language directions. With medium permissibility (middle), C_h is loosened to also accept sentences which are in the top-2 in the source-to-target direction and a top candidate in the opposite direction. In the high permissibility setting(right), this process is symmetric.

2. **Low Permissibility.** Called system P as it is supposed to favor extraction *precision*. It uses the same methodology as the threshold independent approach, but uses both representations $S = C_h$ *and* $S = C_e$. Only pairs that have been matched during filtering in both language directions (primary filter) *and* both representation types are accepted to the candidate list. The intersection between representation types can thus be seen as the secondary filter for this *dual-representation* system. As described in 3.1, C_h and C_e have different modeling capabilities and as described in 4.2.3 also evolve differently as training progresses. Thus, the two representation types turn out to be complementary and this further restriction allows to get rid of the threshold, and the sentence selection becomes parameter-free.

3. **Medium Permissibility.** The combination of representations as in system P is a key point for a threshold-free method, but the final selection becomes quite restrictive. In order to increase *recall* (thus named system R), a more permissive way of selecting pairs should be considered. Instead of taking only the highest scoring target sentence for each source sentence, the top-n are selected (all model R systems in our experiments use $n = 2$). Both representation types are still used. The number of candidates considered is only extended for $S = C_h$ since it is the most restrictive factor at the beginning of training.

4. **High Permissibility.** Generalization of the previous strategy where the method is symmetric in source-target in $S = C_h$ and $S = C_e$. See figure 3.2 for an illustration of the different levels of permissibility when filtering.

Chapter 4

Experiments

In this section, the experiments performed to explore different versions of our online parallel sentence extraction system in a variety of situations (unsupervised vs. semi-supervised) and applications (low-resource MT and filtering) are presented. Apart from reporting the usual translation results, we also perform some in-depth analysis of the representations and the extraction itself, as well as a qualitative analysis of a sample document.

4.1 Data

We use a wide variety of data for extraction, NMT and embedding training, as well as NMT development and testing.

4.1.1 Parallel Corpora

English-French: For all English-French models in experiments I and II, the English-French version of *newstest2014* and *newstest2013* is reserved for **testing**, while *newstest2012* is used for **development**. At training time, a pre-processed version (see 4.1.4) of the development data is used. The test data is not preprocessed and used *as is*, except for a tokenized version to calculate tokenized BLEU scores. As can be deduced from the name, *newstest* corpora are of the news domain. For this, news articles of a source language have been manually translated into the target language.

In order to create more complex pseudo-comparable corpora for the control experiments —which are further described in 4.2.3 and 4.3.3— English-French *europarl* (EP) is preprocessed and used as a rounded complete version (EP complete) of $1.9M$ sen-

tences. The top $500k$ pairs from EP (EP base) are used as a *base* corpus to explore a different type of initialization. The EP corpus contains parlamentary speeches which are translated into several European languages. It is therefore not in the same domain as *newstest*.

English-German: For the filtering experiments in 4.4.2, the English-German version of *newstest2014* is used for testing and *newstest2012* is used for development. The corpus to be filtered is the ParaCrawl corpus (BiCleaner v. 3.0). This is a corpus based on parallel data that has been crawled from the web and therefore contains both valuable and less valuable pairs for NMT training, all while covering a variety of domains.

English-Gujarati: For the English-Gujarati experiments in 4.4.1, we use the concatenation of several parallel corpora. Firstly, the *bible* corpus[1] is a multilingual corpus of bible translations. Further, two corpora specially made for WMT2019[2] are used, namely a crawled corpus (WMT19 Crawl) and a localisation corpus extracted from OPUS[3] (WMT Localisation). Lastly, the *Translation Quality Estimation* (TQE) dataset for Indian languages (Nisarg et al., 2018), which is basically the concatenation of two corpora by the *Indian Languages Corpora Initiative*, which focus on the health and tourism domain each. For development, we use the first 999 sentences from the English-Gujarati version of *newsdev2019*, and for testing we reserve the last 999 sentences. Further, we also report results on the final *newstest2019* corpus.

The final sizes of the data sets can be seen in 4.1.

4.1.2 Wikipedia

The main objective of the following experiments is to explore online sentence extraction on non-parallel corpora. For this purpose, non-parallel corpora are needed. Monolingual corpora exist in abundance in the form of monolingual web-content. However, in order to calculate the margin-based score over a pair of monolingual corpora, all possible sentence combinations must be taken into account. To avoid the $n \times m$ explosion of possible combinations of sentences, where n is the number of sentences in L1 and m in L2, the usage of **comparable corpora** —where aligned documents are of similar topics— is preferable. With topic-aligned documents, the complexity of the search space is reduced to $\sum_{s,t \in D} n_s \times m_t$, where $D = S, T$ are the topic-aligned documents. That is, only all possible source-target sentence combinations within two aligned documents are considered. This way, all the parallel sentences in non-linked documents are lost, but a major gain in speed is achieved.

Wikipedia (WP) is a popular source for comparable documents. In order to use

[1] `http://christos-c.com/bible/`
[2] `http://www.statmt.org/wmt19/translation-task.html`
[3] `http://opus.nlpl.eu/`

	L1/L2 sentences	L1 tokens	L2 tokens
EP complete *en2fr*	1.900.000	56.029.854	61.603.298
EP base *en2fr*	500.000	12184755	13.293.730
ParaCrawl BiCleaner v.3.0 *en2de*	31.078.104	785.158.762	758.536.137
Bible *en2gu*	7.807	260.859	265.188
WMT19 Crawl *en2gu*	10.650	272.559	193.804
WMT19 Localisation *en2gu*	107.637	1.101.131	1.025.859
TQE *en2gu*	50.000	1.213.204	1.055.673
newstest2012 *en2fr*	3.003	80.044	90.027
newstest2012 *en2de*	3.003	72.929	72.603
newstest2013 *en2fr*	3.000	64.807	73.659
newstest2014 *en2fr*	3.003	71.139	81.095
newstest2014 *en2de*	3.003	67.612	63.073
newsdev2019 dev *en2gu*	999	25.866	23.072
newsdev2019 test *en2gu*	999	29.800	27.148
newstest2019 *en2gu*	996	24.046	21.673
newstest2019 *gu2en*	1.016	15.380	17.885

Table 4.1: Size of the parallel corpora in number of sentences and tokens. The values for both *newstest2013*, *newstest2014*, *newsdev 2019 test* and *newstest2019* corpora are calculated from the tokenized version used for evaluation.

it for the following experiments, the WP dumps[4] for English (*en*), French (*en*) and Gujarati (*gu*) are downloaded and later pre-processed as described in 4.1.4.

WP dumps are used for two different purposes: (*i*) to calculate initial word embeddings and (*ii*) as a corpus to extract parallel sentences from and train the NMT systems. In the first case, the complete WP monolingual editions are used. In the second case, only the subset of articles that can be linked across languages using Wikipedia's *langlinks* are extracted. That is, an article is only taken into account if there is the equivalent article in the opposite language. For these purposes, we use `WikiTailor` (Barrón-Cedeño et al., 2015)[5] to get the intersection of articles of both languages.

We additionally use a *en–gu* WP *reference* which was automatically extracted for the 2019 WMT news task. As it is an automatically extracted corpus, many parallel sentences from the original *en–gu* linked WP articles may not be included in this corpus. However, it suffices as a subset of the parallel data in the linked WP to compare our extraction method with.

[4]WP editions downloaded from `https://dumps.wikimedia.org/`. The data for WP *en*, *fr*, *en–fr* is based on a 2014 dump, while *gu* and *en–gu* uses one from 2019.

[5]`https://github.com/cristinae/WikiTailor`

	L1		L2	
	Sentences	Tokens	Sentences	Tokens
WP Edition *en*	92.267.566	2.247.575.834	–	–
WP Edition *fr*	26.595.585	652.265.253	–	–
WP Edition *gu*	4.280.531	74.895.331	–	–
WP Comparable $en - fr$	12.288.721	318.201.381	8.027.123	207.049.147
WP Comparable $en - gu$	546.924	15.483.274	143.120	3.100.463
WP Reference $en - gu$	18.033	415.831	18.033	520.680

Table 4.2: Size of the WP editions, comparable corpora and *en-gu* reference.

	en	
	Sentences	Tokens
NewsDocs *en*	176.220.479	4.012.454.770
NewsDocs *de*	220.443.585	4.426.695.303
CommonCrawl *gu*	3.729.406	67.426.200
NewsCrawl *gu*	244.919	3.341.161

Table 4.3: Number of sentences and tokens of monolingual data after preprocessing.

The final sizes of the WP-based corpora can be observed in table 4.2.

4.1.3 Monolingual Corpora

The following monolingual corpora were used mainly as additional data for training word-embeddings in *en*, *de* and *gu*. For English and German, we use the concatenation of *NewsCrawl* from 2014, 2017, and 2018 (NewsDocs), which is a collection of monolingual news articles. For Gujarati we use the 2018 version of *NewsCrawl* and *CommonCrawl*, which is a crawled corpus of various contents. The final sizes of the corpora after preprocessing can be seen in table 4.3.

To further increase the available data size for training Gujarati embeddings as well as to add similar content to the English word embeddings to be mapped with Gujarati, additional Gujarati news pages were crawled. This yielded an increase of about 2M monolingual Gujarati sentences. While crawling for the news articles, articles written during the period from which the test corpus *newstest2019* was created [6] were not included in the creation of these data sets. The number of sentences and tokens extracted from each news outlet is shown in table 4.4.

[6] September-November 2018

Sentences	Tokens	
Divya Bhaskar *gu*	563.072	17.759.753
News18 *en*	460.097	17.453.729
News18 *gu*	193.455	5.177.957
Gujarat Samachar *gu*	121.349	3.521.586
Sandesh *gu*	892.196	24.102.572
Zeenews *en*	466.449	17.273.255
Zeenews *gu*	244.191	6.624.527

Table 4.4: Number of sentences and tokens of news articles crawled from various news outlets.

4.1.4 Preprocessing

English/French/German: All English, French and German corpora (excluding the evaluation corpora) undergo the same pre-processing. After being sentence split, the corpora are normalised, tokenised and truecased using standard Moses scripts (Koehn et al., 2007). A byte-pair-encoding (Sennrich et al., 2016b) of $100k$ merge operations trained jointly on *en-fr* or *en-de* data respectively is applied accordingly. Duplicates are removed and sentences with more than 50 tokens are discarded. In order to enable a multilingual setup, language tokens indicating the designated target language are prepended to each source sentence. For example, as the English-French setting is bilingual, this simplifies to each French sentence starting with the language token `<en>`, and each English sentence with `<fr>`.

Gujarati: Gujarati corpora are normalized and romanized using the `Indic NLP Library`[7]. The romanized corpora are then tokenized using Moses. As the romanization is case sensitive, no true-casing is performed. At the end, a BPE of $40k$ merge operations trained on both English and Gujarati data is applied to all *en* and *gu* corpora involved in experiments IV.

4.1.5 Cross-Lingual Embeddings

For all unsupervised experiments, we initialze the models using *cross-lingual embeddings*. These are trained using monolingual data only.

The initial monolingual embeddings (of size 512) are trained using `word2vec`[8]. The two embeddings are then projected into a common multilingual space using `vecmap`[9]

[7] `https://github.com/anoopkunchukuttan/indic_nlp_library`
[8] `https://github.com/tmikolov/word2vec`
[9] `https://github.com/artetxem/vecmap`

(Artetxe et al., 2017) . We extract all numerals that occur in both monolingual corpora in order to supply a small seed dictionary for training that is not linguistically motivated. After having projected the embeddings into the same space, they are merged into a single cross-lingual embedding. Whenever a word in the two languages is a homograph, one of the two was chosen randomly.

4.2 Experiments I: Unsupervised Learning

The following experiments make up the core work of this thesis project. They have been designed to explore various components that make up the online-extraction framework: the two scoring functions, different combinations of the two representation types and the filtering technique. All experiments are performed using LSTM's and transformer models. This results in a total number of 10 models, whose specifications are described in detail in the following section.

All models in this set of experiments are initialized using **pre-trained cross-lingual word embeddings** trained on the *en* and *fr* WP editions as described in 4.1.5.

Having initialized the NMT systems with the pre-trained word embeddings, the linked English-French WP articles (WP Comparable) are used as a comparable corpus to extract and train from. Only source-target combinations within two linked articles are considered. Since the number of available WP corpora for English and French is large, articles shorter than 15 sentences are excluded from training to avoid the statistical uncertainty issue described in 3.2. This results in $12M$ and $8M$ of available sentences for English and French respectively, as reported in table 4.2. However, in a low-resource case, it is advisable to cluster smaller corpora into lots of at least 15 sentences, in order not to lose the parallel sentences contained in them.

4.2.1 Model Specifications

The architecture described in 3 is implemented on top of the Open-NMT toolkit (Klein et al., 2017) both for RNN and Transformer encoders. In order to explore the different scoring functions, representation types and filtering techniques, the following models are trained:

- **LSTM**: 1-layer bidirectional encoder with LSTM units, additive attention, 512-dimensional word embeddings and hidden states, and an initial learning rate (λ) of 0.5 with SGD.
  - $\textbf{LSTM}_{\textbf{simP}}$: C_e and C_h are both used as representations in the low permissibility mode. $\text{sim}(S_{\text{L1}}, S_{\text{L2}})$ is used as the scoring function.

- **LSTM$_{\mathbf{margP}}$**: C_e and C_h are both used as representations in the low permissibility mode. margin($S_{\mathrm{L1}}, S_{\mathrm{L2}}$) is used as the scoring function.
- **LSTM$_{\mathbf{margR}}$**: The same as $LSTM_{\mathbf{margP}}$ but C_h is used in the medium permissibility mode.
- **LSTM$_{\mathbf{margH}}$**: C_h is used as the only representation type. It uses the low permissibility mode and margin($S_{\mathrm{L1}}, S_{\mathrm{L2}}$) for scoring. The extraction threshold is set to 1.002 and is determined empirically after observing the margin-based scores of a small portion of the training corpus.
- **LSTM$_{\mathbf{margE}}$**: The same as $LSTM_{\mathbf{margH}}$, but where C_e is used as the only representation type. The threshold is set to 1.3.

- **Transformer**: 6-layer encoder-decoder with 8-head self-attention and 2048-dim hidden feed-forward layers. Adam optimization with $\lambda = 2$ and $beta2 = 0.998$; *noam* learning rate decay (as defined in (Vaswani et al., 2017)) with 8000 warm-up steps. Labels are smoothed ($\epsilon = 0.1$) and a dropout mask ($p = 0.1$) is applied. As is common for transformers, position encodings and *Xavier* parameter initialization (Glorot and Bengio, 2010) are used. All sub-models described in the LSTM category (simP, margP, margR, margH, MargE) also exist for the transformers. As for the single representation models that require a threshold, these have been set to the following values.
  - **Transformer$_{\mathbf{margH}}$**: $\theta = 1.01$
  - **Transformer$_{\mathbf{margE}}$**: $\theta = 1.0$

4.2.2 Results and Discussion

The final **translation quality** of the models described in the previous section can be observed in table 4.5. The models are compared to current state-of-the-art unsupervised NMT and NMT+SMT systems.

The best performing model is Transformer$_{\mathrm{margP}}$, which with a BLEU score of 29.21 for $en2fr$ and 27.36 for $fr2en$ outperforms all other unsupervised NMT models by about 12 BLEU points and more. Also when comparing it to more complex unsupervised NMT-SMT systems such as Lample et al. (2018c), its translation quality on *newstest2014* is comparable. This is despite training on an out-of-domain corpus only 4.7% of the size of the in-domain *NewsCrawl* corpus used to train the unsupervised NMT-SMT system. Nevertheless, Lample et al. (2018c) use monolingual data only. Whether it is easier to collect several hundred millions of monolingual sentences or some ten million sentences within comparable documents really depends on the language combination, domain and task. Therefore, both systems should be considered as being the better choice for distinct situations.

Reference	Corpus, $en{+}fr$ sent. (in millions)	BLEU $en2fr$	BLEU $fr2en$
Unsupervised NMT			
Artetxe et al. (2018b)	NCr13, 99+32	15.13	15.56
Lample et al. (2018a)	WMT, 16+16	15.05	14.31
Yang et al. (2018)	WMT, 16+16	16.97	15.58
Experiments I			
$\text{LSTM}_{\text{simP}}$	WP, 12+8	10.48	10.97
$\text{LSTM}_{\text{margE}}$	WP, 12+8	13.71	14.26
$\text{LSTM}_{\text{margH}}$	WP, 12+8	21.50	20.84
$\text{LSTM}_{\text{margP}}$	WP, 12+8	23.64	22.95
$\text{LSTM}_{\text{margR}}$	WP, 12+8	20.05	19.45
$\text{Transformer}_{\text{simP}}$	WP, 12+8	25.21	24.96
$\text{Transformer}_{\text{margE}}$	WP, 12+8	27.33	25.87
$\text{Transformer}_{\text{margH}}$	WP, 12+8	24.45	23.83
$\text{Transformer}_{\text{margP}}$	WP, 12+8	**29.21**	**27.36**
$\text{Transformer}_{\text{margR}}$	WP, 12+8	28.01	26.78
Unsupervised NMT+SMT			
Artetxe et al. (2018a)	NCr13, 99+32	26.22	25.87
Lample et al. (2018c)	NCr17,358+69	28.10	27.20

Table 4.5: BLEU scores achieved by the unsupervised *en-fr* models on *newstest2014* calculated with `multi-bleu.perl`. Training corpora differ by various authors: News Crawl 2007–2013 (NCr13), 2007–2017 (NCr17), the full WMT data and Wikipedia (WP).

As for the models of this experiment, Transformer models generally outperform their LSTM counterparts, which is to be expected. Also, margP models outperform margR models. This may be surprising given the idea that margR is more permissive in its filtering and should thus find more pairs to train on. The reasons behind this final outcome are therefore discussed in detail in the following sections focusing on extraction and sentence representations.

Both single representation models margE and margH have different outcomes for transformers and LSTMs. While margE outperforms margH for transformers, this is inverted for LSTMs. However, as these models are dependent on setting a threshold manually, there is a large variance on what the translation performance of a margE or margH model can be. It is therefore not possible to say that either one of the two is superior to the other, as they are simply not directly comparable. However, what we can see from this is that single representation models *can* perform on similarly high levels as dual-representation models —such as margR or margP— if their threshold is set

System	Unique Accepted Sentence Pairs LSTM	Transformer
MarginP	1.310.920	2.021.198
MarginR	1.198.977	2.317.057
MarginE	432.314	2.393.147
MarginH	3.002.861	4.893.424
SimP	654.349	743.141

Table 4.6: Total number of unique sentences extracted during training of each unsupervised *en-fr* model.

appropriately (see for example $\text{LSTM}_{\text{margH}}$ and $\text{LSTM}_{\text{margR}}$). However, the threshold exploration is time consuming and it is not guaranteed that an optimal threshold is found.

Lastly, there is a large divergence between $\text{LSTM}_{\text{simP}}$ and its transformer counterpart. While the LSTM version is far behind most other LSTM models applying the margin-based scoring function, $\text{Transformer}_{\text{simP}}$ even outperforms $\text{Transformer}_{\text{margH}}$. This is a surprising result and will be discussed in detail in the next section.

4.2.2.1 Unique Accepted Sentence Pairs and Similarity Distributions

The extraction capacities differ with the model architecture (LSTM or transformer) and technique. When examining the number of *unique* sentence pairs a model extracts over the epochs until convergence, certain tendencies can be observed (see 4.6).

- **Transformers vs. LSTM:** Firstly, transformers consistently extract more unique sentence pairs during the course of training. This may be due to their generally higher **modeling capacities**, which also reflects in their generally superior performance in translation.

  When examining in figures 4.2 and 4.3 how the resulting similarity scores of all rejected and accepted sentences are distributed over the epochs, and comparing LSTM and Transformer models, we can see that the LSTM models do not alter their score distributions as drastically as transformer models do. That is, as training progresses, the LSTM changes and adapts its representations less and much slower. The transformer, on the other hand, generally learns to push rejected sentence pairs apart, while accepted sentences move towards each other. This can be seen at the growingly gentle slopes on the opposing sides of the accepted-rejected spectrum. This trend can also be observed in figure 4.1, where many LSTM models never reach the same difference in mean scores as their transformer counter parts

(see figures 4.1b, 4.1d, 4.1e). This underlines the observation that transformers learn stronger representations that are useful for parallel sentence extraction.

- **Margin-based vs. cosine:** Further, models that use the margin-based scoring function $\text{margin}(S_{\text{L1}}, S_{\text{L2}})$, generally extract significantly more than those models that use cosine similarity (SimP). This can be explained by the **problem of scale** that cosine similarity faces (see 3.2), which makes $\text{sim}(S_{\text{L1}}, S_{\text{L2}})$ less reliable for the task of identifying possible translations. Therefore, it accepts enough false positives at an early stage during training, leading to degraded representations and thus ever less reliable extraction decisions in the future.

  This can very well be observed in figure 4.1e. While mostly all other models learn to increase the accepted and rejected distributions by representing accepted pairs closer to each other and rejected pairs further from each other as training progresses, $\textbf{LSTM}_{\text{simP}}$ does the opposite. At the beginning, there is a clear difference in the average cosine similarity of accepted and rejected pairs, but as the epochs pass and the representation quality decays, the distinction between the average cosine similarity of accepted and rejected sentences decreases. This can be observed in more detail in figure 4.3d, where both rejected and accepted pairs become increasingly similar, finally concentrating almost all pairs at very high cosine similarities. Quite opposite to its transformer counterpart, which decreases its similarity scores. All of this makes the extraction decision less and less reliable, finally leading to a low overall BLEU score for $\text{LSTM}_{\text{simP}}$.

  On the other hand, $\textbf{Transformer}_{\text{simP}}$ finds more sentences than its LSTM counterpart before converging. Interestingly, it does not have the problem of enclosing accepted and rejected distributions, but it does take significantly longer to start increasing the difference in average similarity between the two distributions (epoch 8 vs. epoch 3-4 as in most transformer models applying the margin-based score). This can be observed in more detail in figure 4.3c. Here it becomes clear that *both* accepted and rejected sentences become less similar to their candidate target as training progresses, only that accepted pairs tend to become less similar *slower* than rejected pairs. All of this indicates that the internal representations are affected by the constant false positives that may enter during training due to the error-prone cosine similarity, such that they inhibit accepted pairs from growing closer to each other.

  However, the number of true positives is not overwhelmed by the false positives and enough useful pairs are accepted to slowly adapt the representations to the corpus domain, leading to a steady increase of accepted pairs. This is quite different from models applying margin-based scoring, in which accepted source-target pairs do tend to become closer to each other, due to the smaller amount of false positives. This is also reflected in the significantly higher BLEU score of $\text{Transformer}_{\text{simP}}$ compared to its LSTM counter part. It shows comparable results to other Transformer models that apply the margin-based scoring function, despite it taking much

longer to converge.

- **Word-Embeddings only:** $\text{LSTM}_{\text{margE}}$ is the only model that collects less unique pairs during training than the $\text{LSTM}_{\text{simP}}$ model. However, for all single representation models it needs to be said that their performance in both extraction and translation heavily depend on the **hard-threshold** that was initially set for the extraction decision.

  In the case of $\text{LSTM}_{\text{margE}}$, one might argue that the threshold of 1.3 was set too high. When regarding the comparatively high but stagnant relevance of the threshold in figure 4.4d, this suggests that in fact many sentences that pass primary filtering —via the intersection of both language directions—, are then rejected by the threshold. However, when observing figure 4.2f, we can see that the threshold cut-off appeared before the similarity distribution of accepted sentences reaches its peak. This suggests that using filtering via the intersection of both language directions *without* the cut-off at 1.01 would not have yielded a major increase in accepted sentences.

  We could deduce that LSTM embeddings are less informative for our extraction task than transformer embeddings, since the transformer counterpart did reach a reasonably large amount of $2.4M$ unique extracted sentences despite using embeddings only. Nevertheless, also when looking at 4.1c, both $\text{Transformer}_{\text{margE}}$ and $\text{LSTM}_{\text{margE}}$ end up with the lowest difference in mean scores between accepted and rejected pairs (approx. 0.15 vs. 0.25 or 0.3 for all other models). That is, the pre-trained embeddings are not altered as strongly over the epochs as hidden-state representations; they are rather **rigid representations**. All of this suggests that —while transformer embeddings are superior to LSTM embeddings for our purposes— using these as the sole information for the extraction decision is not optimal.

  Hidden-States only: Both models using hidden-states as their only representation type accept large numbers of unique pairs ($3M$ for $\text{LSTM}_{\text{margH}}$ and almost $5M$ for $\text{Transformer}_{\text{margH}}$).

  When looking closely at figure 4.3a, we can see that the threshold was set quite high for $\textbf{Transformer}_{\text{margH}}$, because the curve of accepted pairs from an early epoch is cut off where it has already descended most of its trajectory. That means, that the major part of the distribution of sentences that would have been accepted without a threshold was cut-off. This leads to a very small number of accepted sentences in earlier epochs, which can also be seen in figure 4.1d. As such, while $\text{Transformer}_{\text{margH}}$ does not accept many sentences in the first two epochs due to the high threshold, after accepting a few more pairs in epoch 3, the representations have adapted enough to make a major jump of $+2M$ unique accepted sentences in epoch 4. This is interesting, because it shows that the model was able to overcome the high threshold once the hidden-representations have adapted towards the few sentences it trained on from the corpus, leading to a **tipping point** in training

where the model is suddenly able to accept more and more sentences.

This may be due to the fact that hidden-representations are randomly initialized at the start of training, carrying only very distorted knowledge of the pre-trained word embeddings in them. As their initial values are closer to zero than the already trained word embeddings —which might carry also extreme values— they are easier to alter and adapt to the training data; they are **flexible representations**. However, since not all accepted pairs are true positives, adapting quickly on them and accepting increasingly more —possibly useless— sentence pairs results in a decreased translation performance, which can be observed in table 4.5.

The same holds for $\textbf{LSTM}_{\text{margH}}$. Only that here the initial threshold of 1.3 was not set high enough. This can be seen at the comparatively high amount of accepted sentences already in its first epoch (see figure 4.1d), which almost reaches $2M$, surpassing all other models at the same point in training. This low threshold shows its effect quickly, as too many false positives are accepted and the representation quality is decayed, leading to enclosing accepted and rejected similarity distributions.

All in all, a careful selection of the threshold is of course advised. Nevertheless, as $\text{Transformer}_{\text{margH}}$ has shown, setting demanding thresholds is not a guarantee against false positives entering in abundance, as hidden representations are quite flexible and can push false positives over the threshold once a tipping point has been reached. As such, it is also not optimal to rely on any of the two representation types, C_h or C_e, only, as they are either too flexible (C_h) or too rigid (C_e).

- **Dual representations:** All models using both representation types for filtering and the margin-based scoring function —namely margP and margR— show similar trajectories when examining their number of unique accepted pairs (see figures 4.1a and 4.1b). Firstly, transformers accumulate more unique pairs over the epochs than their LSTM counter parts. Secondly, all of them —earlier (epoch 2 for $\text{LSTM}_{\text{margP}}$) or later (epoch 4 for $\text{Transformer}_{\text{margR}}$)— separate the similarity distribution of their accepted and rejected pairs significantly. All in all, this is a very **beneficial behavior** for extraction and is reflected as well in the relatively high BLEU scores of these models. This is also achieved by combining the flexible hidden-states and rigid word-embeddings, which together lead to strict extraction decisions without a hard-threshold that could be overcome or simply set inadequately.

  However, the difference between margP and margR being their **permissiveness**, it is interesting to observe how the less restrictive model $\text{Transformer}_{\text{margR}}$ does in fact accept more pairs than its high-restrictive counterpart $\text{Transformer}_{\text{margP}}$. However, the additional extracted sentences also contained more false positives, which can be seen in the slightly lower BLEU score of $\text{Transformer}_{\text{margR}}$. Given that in the margR setting, the rigid word embeddings —which seem permissive at the beginning of training due to their prior knowledge through pre-training— can choose at most one top candidate during filtering, while the flexible hidden-states

—somewhat restricted at the beginning but very permissive later in training— get to choose up to two top-candidates for filtering. One can quickly see how allowing the flexible counterpart to have the larger selection can lead to more false positives. However, instead giving the initially permissive but overall rigid word embeddings the larger choice might not lead to better extraction decisions due to the embeddings comparatively lower capacity to adapt to the data.

4.2.2.2 Intersections: Direction vs. Representation

In the previous section, we have observed how many unique pairs have been extracted by each system and how the representations change based on what is accepted. For a positive change in the representations to happen, the system usually rejects about 99.99% of the combinations it sees and accepts only a very few select pairs to train on. In this section we therefore want to give a special focus on those many sentences that have been rejected.

As described in 3.3, there are different ways of filtering out unwanted sentence pairs. All models use the intersection of the language directions —the *primary filter*— to find pairs of interest, while rejecting all others. As their *secondary filter*, some models —namely dual-representation models— additionally apply an intersection between the two representation types C_e and C_h to filter pairs, while others —single representation models— use a hard-threshold.

In figure 4.4, we can observe the percentage of sentences rejected by the primary filters and the secondary filters of dual and single representation systems of both transformer and LSTM architectures. Naturally, the weighing of the two filters are mirror images of each other: When the secondary filter gains relevance, the percentage of rejections performed by the primary filter goes down proportionally and vice versa. Note that for dual representation models, the data used for visualizing the primary filter in these graphs stems from representation C_h only.[10]

When examining the **dual representation systems** in figures 4.4a and 4.4b, we can see that the patterns of the margP, margR and simP models are quite similar. This shows that threshold-less filtering, using intrinsic features of the models, works in similar ways regardless of the specific architecture of the sequence-to-sequence model. However, for transformer models the impact of the secondary representation seems to be oftentimes significantly higher —for simP for example about twice as high— compared to their LSTM counterparts. This again underlines the observation that transformers de-

[10]This choice was made for dual-representation systems to visualize the logic that in a primary step, the representations C_h are filtered by both language directions, accepting a small minority of combinations. These are then further filtered via the intersection with C_e, which naturally underwent the same primary filtering beforehand, but which is unseen here.

velop more expressive representations that help them discern more and more (un)wanted sentences as training progresses, making them more apt than LSTM representations to filter out sentences that would otherwise be accepted according to the primary filter.

When concentrating on both **margP** models, we can see that the secondary filter reaches a peak at epoch 2. This indicates that epoch 2 is an important event for the extraction in margP models. In fact, in figure 4.1a we have seen that the same epoch is responsible for a major increase of unique accepted pairs. One explanation when regarding figure 4.4a is that by epoch 2, our flexible C_h is starting to adapt to the data, identifying more interesting pairs to accept via primary filtering, while the primary filter in C_e —the more rigid representation— is adapting slower to the new domain, which leads the symmetric difference of the two representations in the secondary filter to be proportionally larger compared to the previous epoch. Thus, the secondary filter gains in relevance in the second epoch due to improved representations in C_h and a conservative C_e. As C_e slowly adapts to the training data, the symmetric difference between the two representation shrinks again slightly as the primary filtering of *both* representation types accept more of the same sentences. This can be observed in the slightly sinking trend of the secondary filter of margP in later epochs.

A similar pattern can be observed in the **margR** models. However, here the relevance of the secondary filter is generally moved up compared to the margP model. Due to the higher level of permissibility in margR, more sentences pass the primary filter of C_h, which then leads to an enlarged symmetric difference to the low-permissibility C_e when performing secondary filtering.

The **simP** models also have similar curves, only that they do not reach their critical point until later during training (epoch 4 for $\text{Transformer}_{\text{simP}}$ and epoch 3 for $\text{LSTM}_{\text{simP}}$ in this case). This is also reflected in the sentence accumulation of both systems (figure 4.1e), where $\text{LSTM}_{\text{margR}}$ makes a leap in numbers of unique accepted sentences during epoch 3, while its transformer counterpart shows a very smooth and slowly growing curve.

All in all, we can say that the rejections in dual representation models work via the **interplay** of its two representations. When C_h quickly adapts to the training data, C_e omits an explosion of accepted sentences due to its more rigid nature and different modeling capacities. This careful behavior in earlier epochs allows for the representations in both C_e and C_h to adapt on less false positives and therefore gives both representation types time to learn relevant features. Once also the more conservative C_e representations have adapted to the training data, the relevance of secondary filtering goes down slightly as both representation types accept and reject similar sentences in primary filtering.

Now, compared to the rather *homogeneous* trends observable among the dual representation models, **single representation systems** seem to be more variable. The one model that especially catches the eye is the transformer of type **margH**, due to its

zigzag of changing relevance of both filters. However, earlier we have already discussed that the threshold for this model was initially too low, but that epoch 4 is marked as the turning point where suddenly about $2M$ new sentences were accepted. One possible explanation may be that by epoch 3, C_h has adapted enough on the training data to show an increase in accordance between its two directions, letting more pairs pass the primary filter. While a good portion of these sentences then also passed the threshold, many of them did not, leading to an increased relevance of the secondary filter and a peak at epoch 3. The newly accepted pairs, however, further altered the representations in such a way, that by epoch 4, many more pairs were able to pass the threshold while pushing down its comparative relevance for rejection. The critical point in epoch 3 can therefore be seen as the *harbinger* of an increased number of accepted sentences passing the threshold in a few train steps ahead.

Model $\text{LSTM}_{\text{margH}}$ on the other hand, shows no significant sign of change in the relevance of both filters. This is due to the threshold being set too low, as discussed earlier in 4.2.2.1, which leads to most pairs that were accepted by the primary filter also passing the threshold, making it close to irrelevant.

In 4.2.2.1 we have also seen that pre-trained embeddings as our C_e adapt slowly to the training data, which is reflected here in the shallow curves of both **margE** models. In figures 4.1c and 4.2e we can observe how $\text{Transformer}_{\text{margE}}$ manages to slowly adapt the representations as the mean difference of the accepted and rejected distributions is increased. In this slow but continuous change, the threshold grows slightly in importance while staying close to stagnant otherwise. The threshold of its LSTM counterpart, however, never truly grows in relevance, simply because it is already set to a large value. This leads to it always rejecting a relatively high share of sentences (0.3%) as compared to other models, leaving the slowly adapting embeddings almost no chance to overcome the threshold and change the comparable relevance of the secondary filter.

To conclude, for the single representation models it can be said that as word embeddings are more rigid, margE models have a rather stagnant change in relevance of the primary and secondary filters, while with the rather flexible representations of margH, the change in relevance can also be similarly idle but also surprisingly dynamic depending on how the threshold was set.

4.2.2.3 Qualitative Examples

In earlier sections we have discussed different quantitative features of the different models trained within the scope of this experiment, and set up some hypothesis as an attempt to explain the connection between different events during training and extraction. Now, to give a more qualitative image of how the extraction performance changes over the epochs, and which role each of the two representation types plays, we will focus on a single

	C_e only	C_h only	$\cap$
Epoch 0	3 (2)	5(2)	1(1)
Epoch 1	0	3(2)	8(8)
Epoch 8	1(1)	0	10(9)

Table 4.7: Sentence pairs accepted from the example WP article by either representation C_e and C_h or their intersection $\cap$. Numbers in brackets indicate how many of the accepted sentences were true positives. Pairs are only counted as true positives if the two sentences are translations of each other *without* and additional information added or missing on either side.

WP article talking about the charming *Leopard Grouper*. The article has been chosen by chance[11]. This qualitative analysis focuses on our best-performing model, namely $\text{Transformer}_{\text{margP}}$. The main focus lies on the three most important points during its training: The initialization phase having seen pre-trained embeddings only (Epoch 0), the start of the second epoch in which extraction suddenly gains momentum as discussed in 4.2.2.1 (Epoch 1) and at the end of training when having reached convergence (Epoch 8).

Looking at the article as a whole, one can say that it is *almost* parallel. However, the French sentences often tend to cover the content of several English sentences or simply add additional information to otherwise parallel sentence pairs, which makes extraction especially difficult. However, the system coped very well with this article, as by the end of training, only one parallel sentence was not identified and only one sentence which was not completely parallel (due to additional content in the French sentence) was accepted nevertheless.

When observing in table 4.7 the number of sentences accepted by either of the two representation types C_e and C_h (primary filters) and their intersection (secondary filter), we can see that in the initialization phase many sentences passed the primary filters, while only one the secondary filter. Especially C_h seems to be keen in accepting pairs which are not necessarily parallel. This again confirms our previous observation that when extracting with pre-trained word embeddings, C_e already has some initial knowledge about the languages involved which helps it identify parallel sentences more precisely (2/3 vs. 2/5 for C_h in this case). Due to its rigid nature, C_e acts as a damper on the false positives accepted by C_h. Already at the end of the first epoch, both representations start to align, extracting more pairs. By the end of the training, the system has found almost all parallel sentences in the article, except for one which fails to be accepted by C_h.

[11] It was the first article pair in the shuffled list provided to the systems for extraction that contained any parallel sentences.

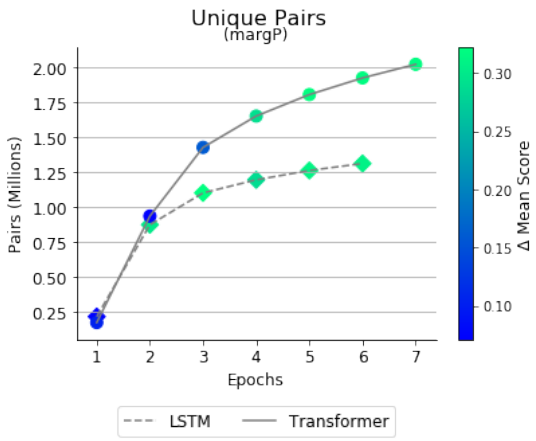
Now, in order to gain some deeper insight into what kind of sentences the two representation types accepted individually or in concordance, we will focus on the few select sentence pairs in table 4.8.

Firstly, all the previously mentioned short sentences in English that have been combined into longer French sentences do not pass any of the primary filters. As can be seen in candidate pair a), not even the longer French sentence that semantically includes the semantics of the short English sentence is accepted by any representation at any point in training. One reason for the models certainty in rejecting this sentence combination might be based on how the sentence representations used are calculated. Instead of using the mean of each time step, the sum is taken, which indirectly adds **length information**. This is because short sequences will on average have sentence representations that contain lower values than those of long sequences. Therefore, this indirect length information might have helped filter out this candidate pair despite the partial semantic overlap.

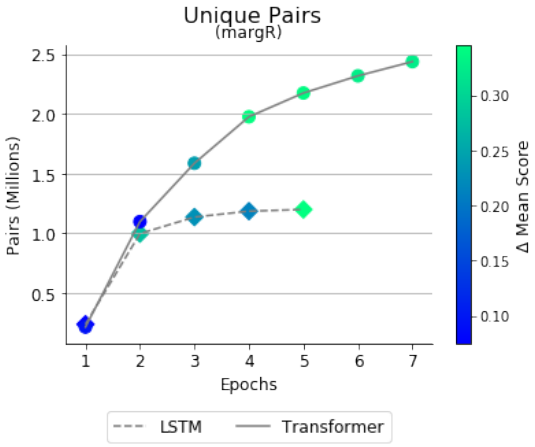
The evolution that is seen in candidate pairs b) and c) is also interesting. Word embeddings trained with `word2vec` might encounter difficulties when dealing with words that modify nouns; so-called **modifiers**. This is because they often occur in similar — possibly broad— contexts (e.g. *a yellow/red/white rose/book/etc.*). This is especially true for words that rely on an extra-linguistic reference scale.[12]. This large amount of descriptive adjectives (*light brown, reddish, light green-gray, small red or dark etc.*) might be the reason why the system rejected both b) and c) with both representations, while most other parallel sentences that were rejected by the secondary filter in epoch 0 at least passed the primary filter of one of the two representations. However, even though the system only accepted one sentence in the first epoch, when starting epoch 2 it already correctly identifies b) as parallel based on what it has learned from other articles. Representation C_h has also adapted enough to the data to accept c), but the rigid embeddings C_e take longer to adapt, allowing the candidate pair to be extracted later in training.

Candidate pair d) is the only pair that was accepted by both representation types from the very beginning. It is therefore an important pair, as it constitutes part of the first epoch that sets the ground work for adapting the representations towards the training data. It is easy to see why this sentence was easy to handle by both representation types: It contains **homographs** that also represent the same meaning in both phrases. The main examples being [*cau@@, ¨ , V*]. It might seem irrelevant at first that some BPE token is shared as well as a double occurance of V. However, as the vocabulary is shared between the two languages and their embeddings initialized based on that vocabulary, having the same tokens appear in a candidate pair has a strong effect on the decision especially at the beginning of training. Similarly, it has also been observed that

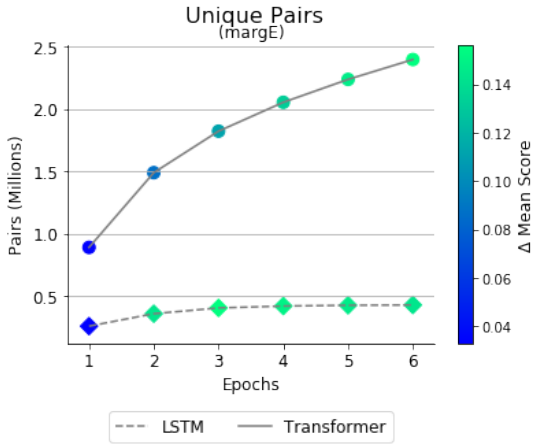
[12]A large mouse is probably still smaller than small giraffe, for example. The scale used to give meaning to the modifiers *large* and *small* depends on the type of entity that is being modified.



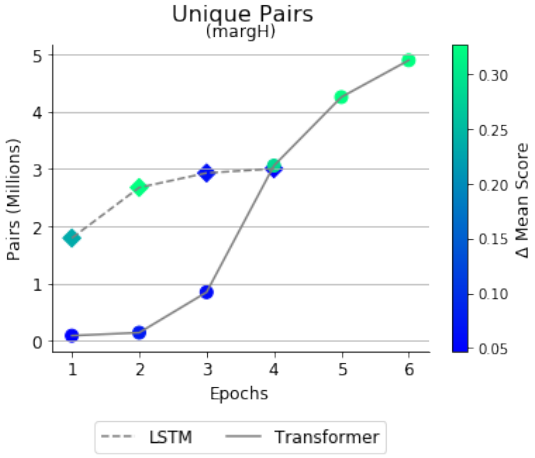
(a) Unique pairs (margP)



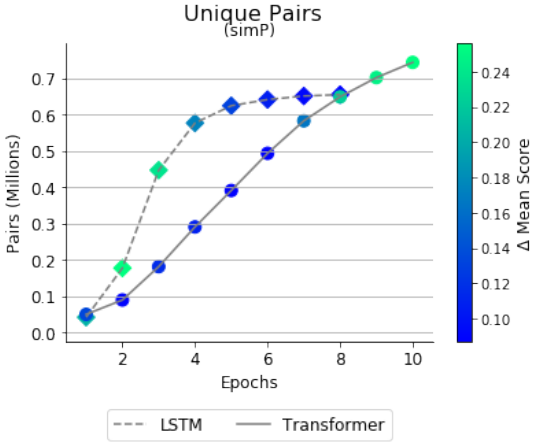
(b) Unique pairs (margR)



(c) Unique pairs (margE)

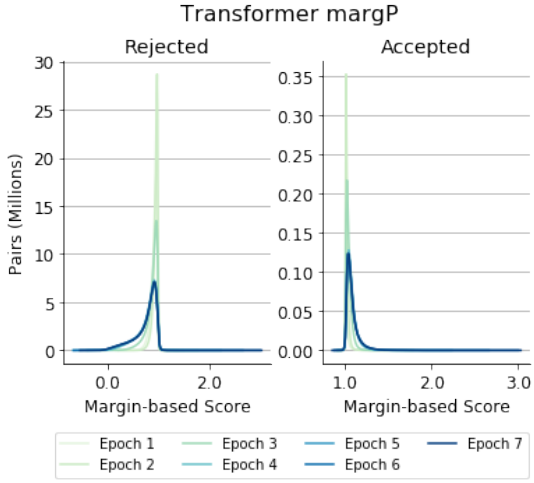


(d) Unique pairs (margH)

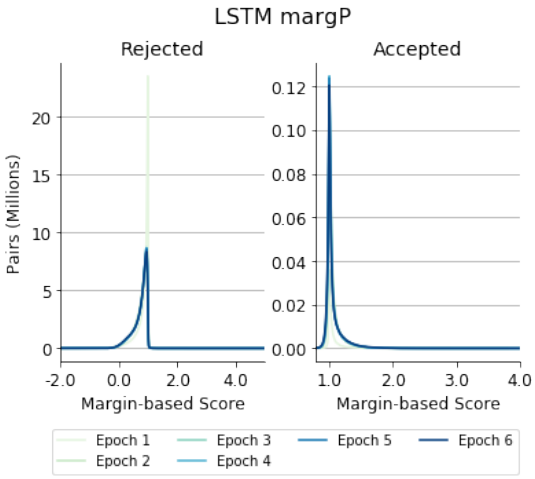


(e) Unique pairs (simP)

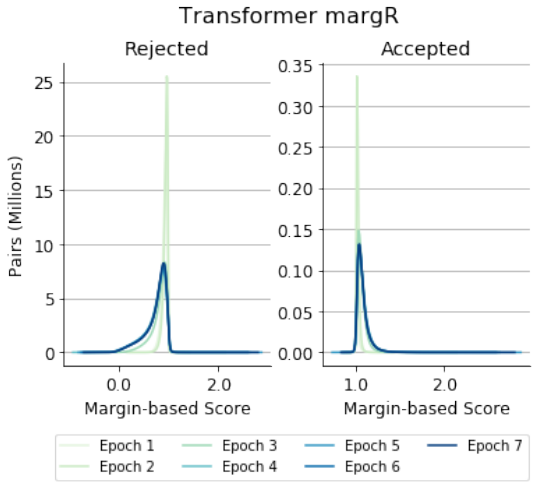
Figure 4.1: The number of unique pairs extracted by each unsupervised *en-fr* system as training progresses. The color map indicates the difference in the average score of accepted and rejected pairs.



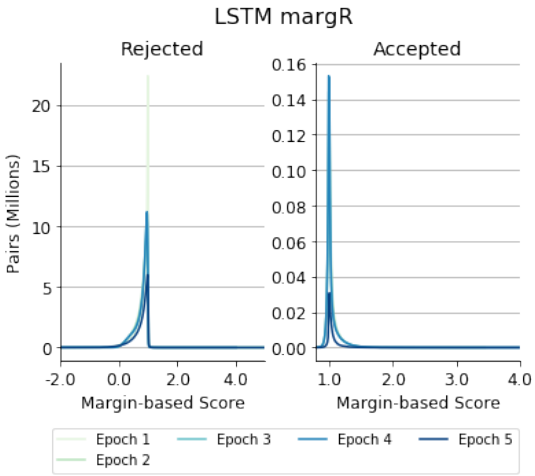
(a) Similarity distribution ($\text{Transformer}_{\text{margP}}$)



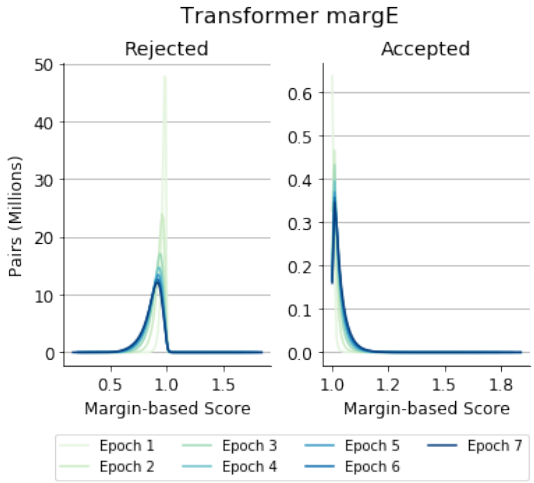
(b) Similarity distribution ($\text{LSTM}_{\text{margP}}$)



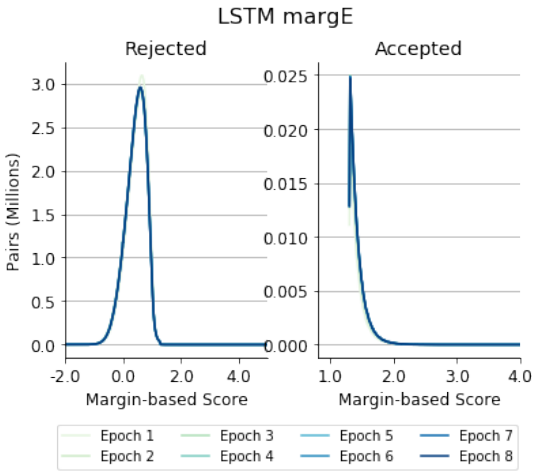
(c) Similarity distribution ($\text{Transformer}_{\text{margR}}$)



(d) Similarity distribution ($\text{LSTM}_{\text{margR}}$)

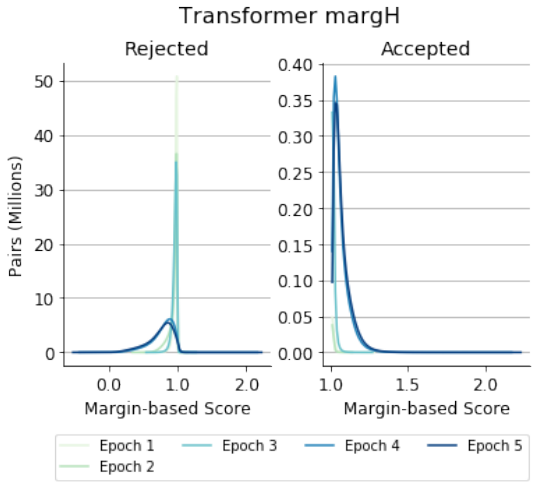


(e) Similarity distribution ($\text{Transformer}_{\text{margE}}$)

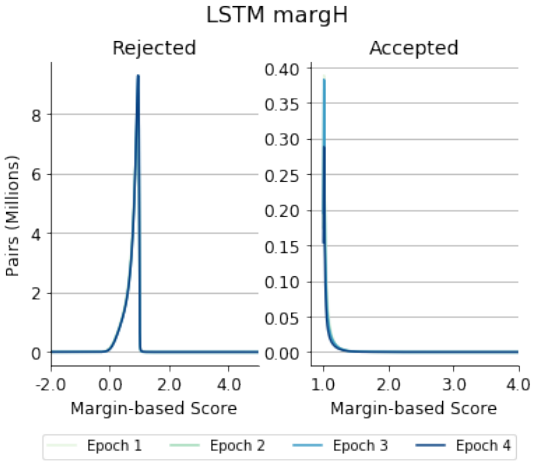


(f) Similarity distribution ($\text{LSTM}_{\text{margE}}$)

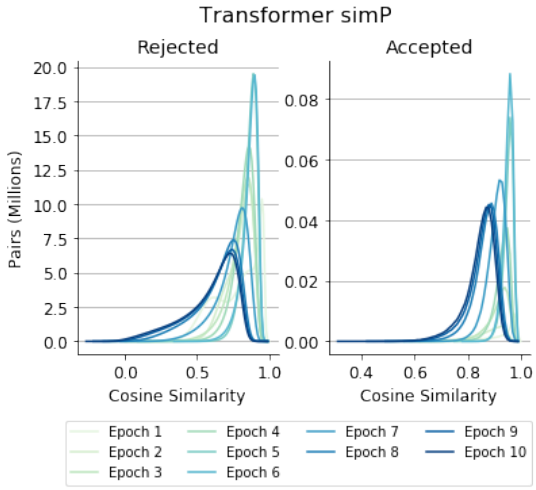
Figure 4.2: Distribution of the similarity score for rejected and accepted pairs when extracting from *en-fr* Wikipedia using different unsupervised models.



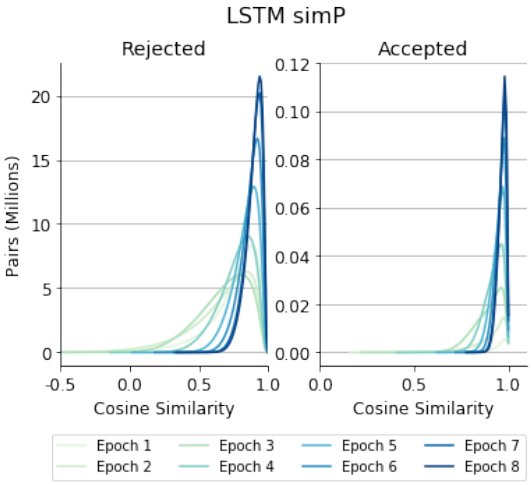
(a) Similarity distribution ($\text{Transformer}_{\text{margH}}$)



(b) Similarity distribution ($\text{LSTM}_{\text{margH}}$)

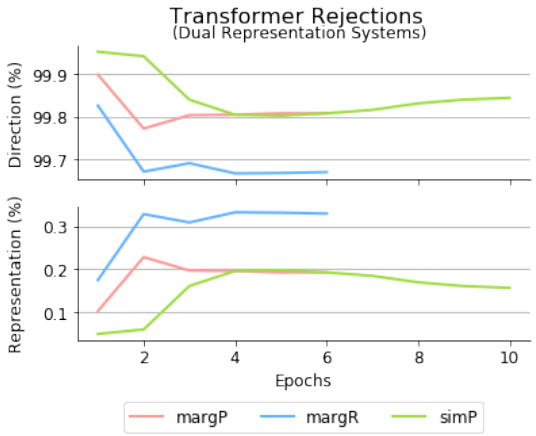


(c) Similarity distribution ($\text{Transformer}_{\text{simP}}$)

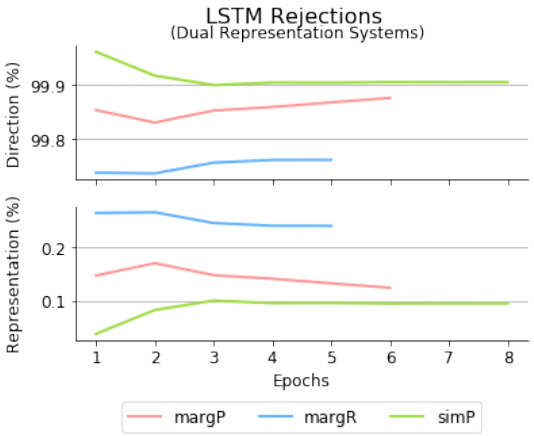


(d) Similarity distribution ($\text{LSTM}_{\text{simP}}$)

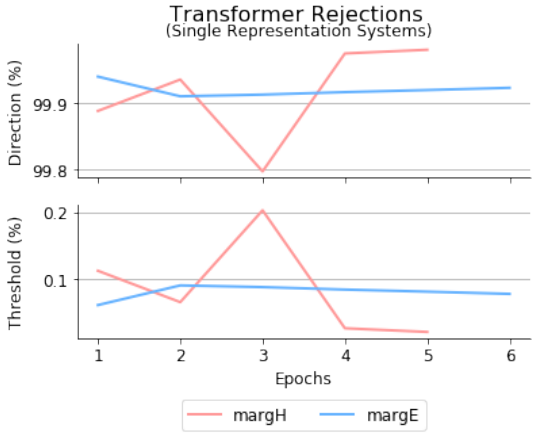
Figure 4.3: Similarity distribution, part II. (Continuation of 4.2.)



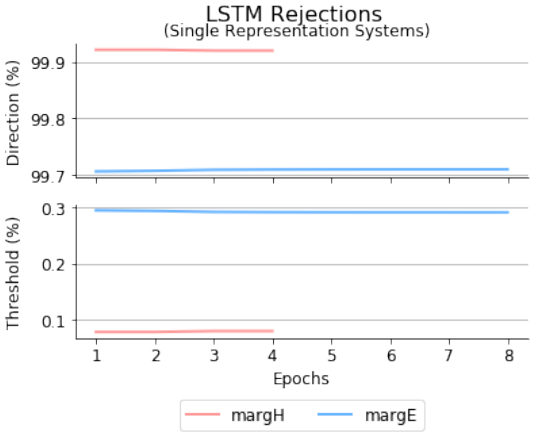
(a) Dual Representation Transformers



(b) Dual Representation LSTMs



(c) Single Representation Transformers



(d) Single Representation LSTMs

Figure 4.4: Percentage of sentence pairs rejected via the source-target direction intersection (top) and the secondary filter (bottom) of the unsupervised *en-fr* models. The secondary filter is further defined as the representation intersection (dual-representation systems) or the hard threshold (single-representation systems). A comparison between transformers and LSTM models is made.

	Candidate Pair	Epoch 0			Epoch 1			Epoch 8			Gold
		C_e	C_h	$\cap$	C_e	C_h	$\cap$	C_e	C_h	$\cap$	
a)	**en**: The mouth is big and has a superior position . **fr**: Corps él@@ ancé , comprimé latér@@ alement , il se termine en pointe , la bouche est légèrement supérieure .	r	r	r	r	r	r	r	r	r	r
b)	**en**: The body background coloration is light brown , reddish or light green-@@ gray . **fr**: La couleur de fond du corps est be@@ ige clair , rouge@@ âtre ou gr@@ is-@@ vert clair avec des mar@@ br@@ ures .	r	r	r	a	a	a	a	a	a	a
c)	**en**: The front sn@@ out is covered with small red or dark dots . **fr**: Le mus@@ eau est const@@ ell@@ é de petits points noirs ou rouges .	r	r	r	r	a	r	a	a	a	a
d)	**en**: The cau@@ dal fin is distinguished by two red to dark lines forming a " V " and another black line parallel to the top line of the " V. " **fr**: La nage@@ oire cau@@ dale se particul@@ arise également par deux traits obli@@ ques de couleur rouge à sombre formant un " V " ainsi que par un trait noir parallèle au trait supérieur du " V " .	a	a	a	a	a	a	a	a	a	a
e)	**en**: Ce@@ phal@@ op@@ hol@@ is le@@ opar@@ dus is carnivorous and its diet consists mainly in small fishes and crustaceans , it 's an ambush predator **fr**: Les points caractéristiques de cette espèce , la différenci@@ ant notamment de Ce@@ phal@@ op@@ hol@@ is uro@@ det@@ a , résident dans la présence de deux taches noires sur la partie supérieure du pé@@ don@@ cule cau@@ dal .	r	a	r	r	r	r	r	r	r	r
f)	**en:** It is proto@@ gy@@ nous hermaph@@ rodite , which means the female can evolved to male during its life . **fr**: Il est her@@ m@@ aph@@ ro@@ dite proto@@ gy@@ ne , c' est-à-dire que l' animal est d' abord femelle à la maturité sexuelle puis devient mâle .	r	r	r	r	a	r	a	r	r	a

Table 4.8: Example sentences extracted by either representation type C_e, C_h or their intersection $\cap$ at the end of epoch 0 (beginning of training), 1 and 8 (end of training) of Transformer$_{\text{margP}}$. An English translation of the French sample sentences can be found under 5 for reference. Note that @@ denotes a sub-word boundary as defined by our BPE encoding.

sentences that contain numbers are especially popular for extraction in the first epoch, simply due to them consisting of the same tokens.

In e) it is interesting to see how C_e acts as the **damper** on the decisions that C_h takes at the beginning of training. As the two candidates, which are in fact not parallel, contain many BPE homographs (*Ce@@, phal@@, op@@, hol@@, is*), these have a strong effect on C_h, as they are treated similarly by the recently initialized encoder. On the other hand, C_e can use its prior knowledge of the words it has seen during pretraining on monolingual data to see that most of the other words in the two sentences are semantically not similar enough to each other. Therefore, the sentence does not pass the secondary filter and in later epochs —once C_h has been adapted to the data— is never considered again.

Lastly, candidate pair f) was a tricky case for $\text{Transformer}_{\text{margP}}$ as it is never accepted by both representations at once, even though it arguably could be considered as a true parallel pair. Reasons for this might be the **diverging BPE** for one of the indicative words (*hermaph@@ rodite* vs. *her@@ m@@ aph@@ ro@@ dite*). Also, while the semantic of the two sentences is very alike, the **phrasing** is more concrete in the French version, which specifies that *l'animal est d'abord femelle à la maturité sexuelle puis devient mâle* [the animal is female at the beginning of its sexual maturity and then becomes male]. In the English sentence this is compressed to *females can evolved* [sic] *to male during its life*. This semantic complexity is still hard to be captured by the representations, as the pairs is sometimes accepted by either one of C_e or C_h but never both.

All in all, the hypothesis that were drawn earlier about the interplay of both representation types (e.g. C_e being more rigid and adapting slower, stopping C_h to accept too many false positives at the beginning of training etc.) are also observed in this concrete example article. Further, we could observe the importance that homographs, such as numerals and a concurring BPE encoding have in the first epoch. That is, how d) was easily identifiable as parallel due to its homographs, and how f) was never extracted possibly due to its diverging BPE of an indicative word.

In fact, when looking at the percentage of tokens accepted by $\text{Transformer}_{\text{margP}}$ that are BPE subword units, numerals or other homographs in source and target (see figure 4.5), we can observe that at the beginning of training these three groups are more strongly represented than later in training. At the very beginning of training, almost every fourth token in an accepted source sentence also appears in the matched target. However, this also includes high-frequency punctuation, with approximately 34.54% (see figure 4.6), which probably are little useful to identify possible parallel sentences since they occur in close to every sentence. Nevertheless, ca. 23.62% are BPE subword units and 6.58% numerals which are informative indicators of whether a sentence pair should be extracted or not. This also explains the larger ratio of BPE subwords and numerals in the first epoch as opposed to later in training. As training progresses, the

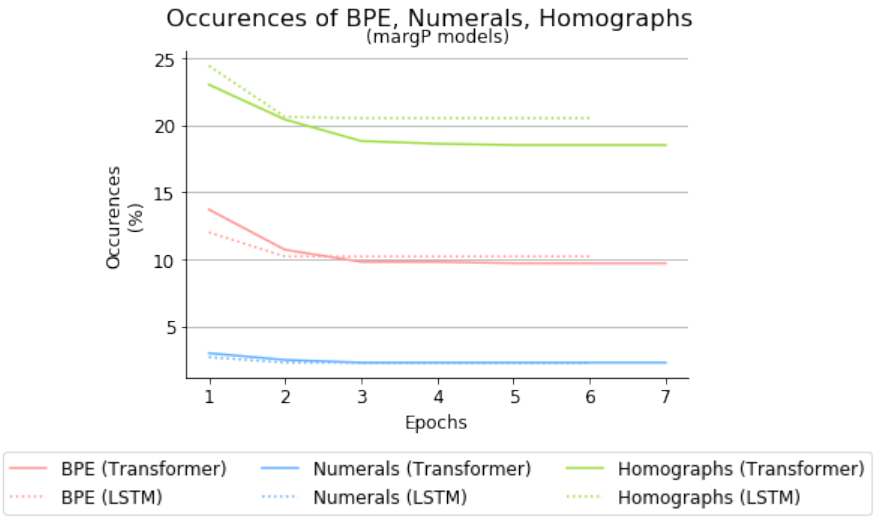


Figure 4.5: Percentage of tokens consisting of BPE subword units, numerals or homographs found in pairs accepted by the two *en-fr* margP models in each epoch during training.

sentence representations adapt to the data and do not rely as heavily on the occurrence of homographs anymore. Another important group of words in the first epoch are named entities, which also make up a large part of the homographs group.

All of this suggests that homographs, such as shared named-entities, a common BPE encoding and shared numerals are an important factor at the beginning of training and extraction. This raises the question of how an unsupervised model as the ones described in this chapter will perform when working with two very distant languages, which do not share many (useful) homographs due to diverging orthography or even scripts. In 4.4.1 we encounter such a setting, where a Gujarati-English model is trained in a very similar setting as the English-French models in this set of experiments.

4.2.3 Control Experiments: Extraction Accuracy

Earlier, the translation accuracy and some of the statistics behind the representations and their roles in extraction have been examined. However, as there is no underlying ground truth of parallel sentences for the English-French Wikipedia, the actual performance of the extraction in terms of precision, recall or f-score can not be calculated. In order to get a vague idea of how the models performance is for parallel sentence identification, we use a pseudo-comparable corpus with similar technicalities as the original WP corpus.

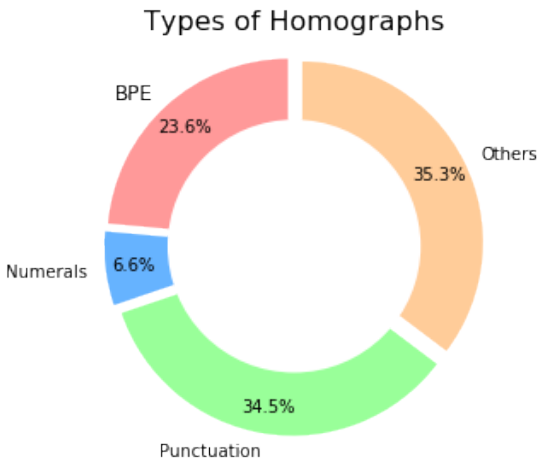


Figure 4.6: Types of homographs found in the accepted pairs extracted by $\text{Transformer}_{\text{margP}}$ during the course of its first epoch. The class *others* consists largely of named entities and common words.

4.2.3.1 Design

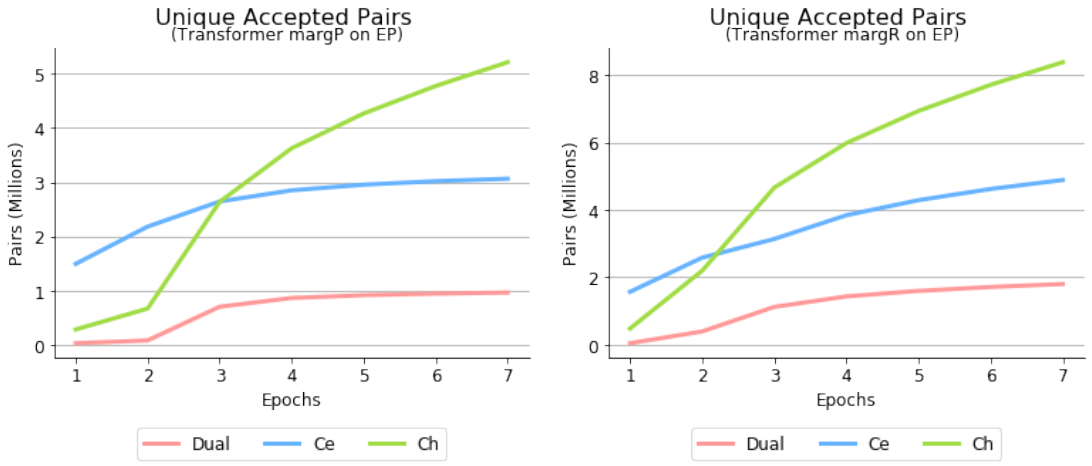
4.2.3.1.1 Pseudo-Comparable Corpus

In order to create a pseudo-comparable corpus, we sample $1M$ parallel sentences from the pre-processed EP complete. The rest of the sentences in EP are over-sampled and then shuffled to create a larger number of negative samples.

The best performing model on WP —$\text{Transformer}_{\text{margP}}$— extracted about $2M$ pairs ($4M$ sentences) from a bilingual corpus comprising approximately $20M$ unique sentences. If we, for simplicity, assume it extracted optimally, 1 out of 5 sentences in *en-fr* WP may have a parallel target pair. This yields a noise ratio of 1:4 parallel-negative samples. This is reflected in the pseudo-parallel corpus by adding $4M$ **negative samples**, which are then shuffled with the true pairs.

The **mean lengths** of WP articles in English and French are 70 and 46 respectively. We therefore split the pseudo-comparable corpus into lots of 46, which means that each lot contains approximately 9 parallel sentences. The English side is then augmented with 24 additional negative samples, to reach a total of 70 sentences. The actual noise-ratio in each lot is therefore much higher than the optimistic case of 1:4, which gives a more realistic setup while reflecting the average WP articles lengths.

While the length and noise statistics might be similar, the contents of the two corpora diverges strongly. WP contains a variety of topics and it is not always as clear whether a sentence pair should be considered parallel or not (as has been observed in section 4.2.2.3). In this pseudo-comparable corpus, however, the topic is parliamentary speech,



(a) Accepted EP pairs by Transformer$_{\text{margP}}$ (b) Accepted EP pairs by Transformer$_{\text{margR}}$

Figure 4.7: Number of unique sentences accepted by the secondary filter (dual) and the single representations C_e and C_h of both models used for the unsupervised *en-fr* control experiments.

and while the topic is always constant, the difference between true and negative samples is more clear.

4.2.3.1.2 Models

We focus on the two best-performing models, Transformer$_{\text{margP}}$ and Transformer$_{\text{margR}}$, also to observe whether margR models can truly be thought of as *high recall but medium precision* and margP as *high precision but medium recall* as introduced earlier. Similarly as before, the models are initialized on the same pre-trained multilingual word embeddings used in the main experiments. The models are then used for joint extraction and training on the pseudo-comparable corpus.

4.2.3.2 Results

We calculate the precision and recall of the two models in two different ways; by epoch (*unique*) and considering the whole training process as a whole (*accumulated*). Where the accumulated scores are given for a specific epoch, the extracted data up to and including that epoch are treated as one corpus and is then compared to the $1M$ ground-truth pairs.

When observing in 4.7 the number of unique pairs that were accepted by the two systems or passed one of their primary filters, it becomes clear that margR does in fact extract significantly more sentence pairs than margP. Therefore, the idea that margR

		$\text{Transformer}_{\text{margP}}$		$\text{Transformer}_{\text{margR}}$	
		Unique	Accumulated	Unique	Accumulated
Dual	Precision	0.96378	0.94692	0.74100	0.73924
	Recall	0.67633	0.95258	0.81882	0.98371
C_e	Precision	0.79015	0.60711	0.45455	0.45848
	Recall	0.72907	0.97874	0.91529	0.99562
C_h	Precision	0.42498	0.42455	0.28461	0.40286
	Recall	0.89028	0.99464	0.85520	0.99389

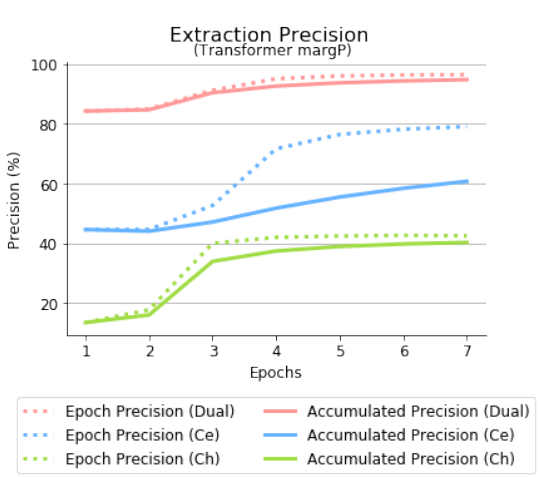
Table 4.9: Unique and accumulated precision and recall of both unsupervised *en-fr* $\text{Transformer}_{\text{margP}}$ and $\text{Transformer}_{\text{margR}}$ models in the final epoch (epoch 7) on the EP-based pseudo-comparable corpus.

is more **permissive** than margP holds, as it almost extracts twice as many pairs. Interestingly, not only C_h, which is set to high-permissive mode and the resulting final extraction are more permissive, but also C_e, which by itself uses the same settings as in margP, accepts significantly more pairs in margR. This may be due to the increased amount of overall accepted sentences, which in return change the representations in C_e at each train step, making them adapt to the additional data.

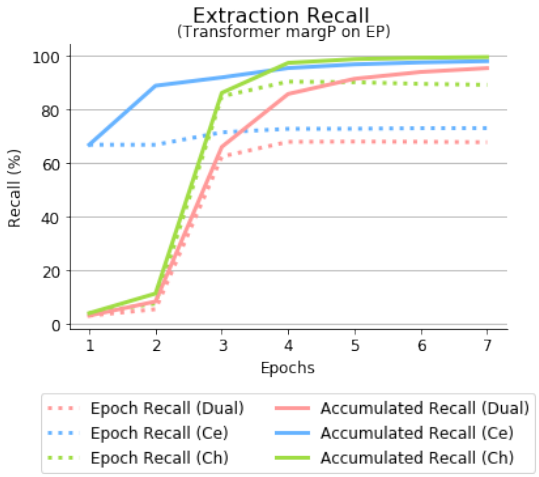
Further, it is interesting to see how C_e starts off being the representation finding the largest amount of pairs to extract, but soon becomes overtaken by C_h. This reflects the earlier observation that C_e, as it is already pre-trained, it finds more pairs in the first epoch, while C_h does not. However, as C_h is randomly initialized and its values close to zero, they quickly adapt to the data and overcome C_e in the amount of pairs it extracts. From now on, C_e works as a damper on the vast amount of sentences entering from C_h. This is the case for both models.

In table 4.9, we can see the unique and accumulated precision and recall for both models in epoch 7, i.e. at the end of training. It can be seen that the accumulated recall is only slightly lower for margP compared to the high-permissive counter part. On the other hand, this slight drawback is overcome by the vast difference in **extraction precision** between the two models. While margP is in fact on the top end of the precision scale, margR is significantly below, accepting a larger amount of false positives during training. This can be closer observed in figure 4.8, where we see how the extraction precision and recall evolves over the epochs for each filter.

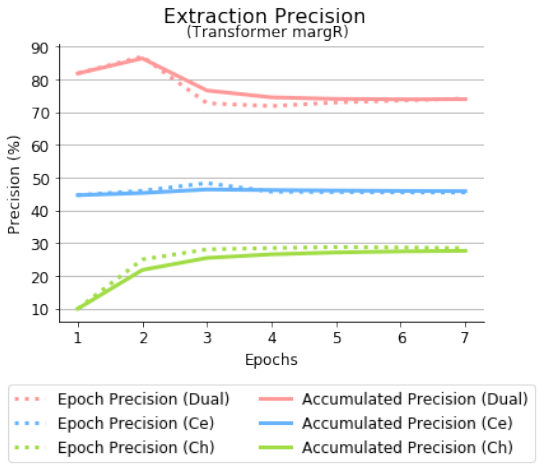
What is most astounding is the fact that each primary filter on its own, C_e and C_h, never reach high precision scores. By taking the **intersection** of the two in the secondary filter (dual), we suddenly reach high levels of precision. This shows that the two representation types do complement each other. This secondary filter does not



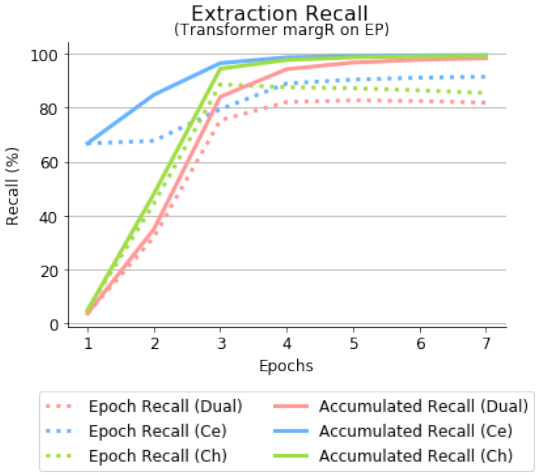
(a) Precision Transformer$_{\text{margP}}$



(b) Recall Transformer$_{\text{margP}}$



(c) Precision Transformer$_{\text{margR}}$



(d) Recall Transformer$_{\text{margR}}$

Figure 4.8: Precision and recall of unsupervised *en-fr* Transformer$_{\text{margP}}$ and Transformer$_{\text{margR}}$ on the pseudo-comparable EP corpus.

come with a significant loss in recall either, as we can see that by the end of training, the accumulated recall for both models closes in to the recall of their primary filters.

We can further observe that the recall is very much correlated with that of C_h. This again underlines the idea that as C_h adapts, more pairs pass both the primary filter C_h *and* the secondary filter. C_e on the other hand already has a relatively high recall from the very beginning and rises more gradually as the epochs pass, converging with C_h and the final extraction by the middle of training.

Lastly, we should note that as the precision of margP steadily increases over time, this is not the case for margR. Even though both primary filters increase in precision, the intersection between the two representations is faulty, leading to a decreasing accuracy

	Precision (%)	Recall (%)
Grégoire and Langlais (2018), 0% noise	99.26	93.50
Grégoire and Langlais (2018), 50% noise	98.32	93.60
Grégoire and Langlais (2018), 90% noise	97.94	95.00
$\text{Transformer}_{\text{margP}}$ with singles	94.69	95.26
$\text{Transformer}_{\text{margP}}$ without singles	**99.78**	**98.15**

Table 4.10: Accumulated precision and recall of $\text{Transformer}_{\text{margP}}$ in its final epoch over the pseudo-comparable corpus with negative samples that do not have translations at all (*with singles*) as well as on an alternative corpus where each sentence has a translation somewhere in the document (*without singles*). These are compared to the extraction performance Grégoire and Langlais (2018) on the same dataset with variable degrees of unaligned sentence pairs.

of the model as we pass epoch 2. One reason for this may be the generally lower precision scores of C_e and C_h when compared to their counterparts in margP. While they start off having very similar scores in the first epoch, they grow much slower in margR as training continues. C_e, for example, seems to barely grow in precision. Due to the larger amount of false positives, the representations become less reliable, and their intersection faulty, which then is reflected in the decreasing final extraction precision in later epochs.

Therefore, it can be said that margP has a major advantage over margR. It reaches high levels of precision with only a minor difference in recall to the high permissive model. Its much lower amount of false positives encountered during training is reflected in its higher BLEU scores, not only in this control experiment, but also in the main experiments on WP.

Further, we can set our extraction performance into context with other approaches by comparing with Grégoire and Langlais (2018). They trained a siamese network on a pseudo-comparable corpus built from *europarl* with varying degrees of incorrectly aligned sentence pairs (*noise*). This differs from our approach, as we do not pre-align any pairs at all, which means that our system sees about 99.9% of *noise* to extract from. Also, our comparable corpus contained sentences which did not have any translation (*singles*) in the document, while this is not a special consideration in Grégoire and Langlais (2018). We therefore also train a smaller $\text{Transformer}_{\text{margP}}$ without singles to present the extraction performance in a more comparable setup.

As can be observed in figure 4.10, our margP model on the smaller pseudo-comparable EP corpus without singles outperforms any of the models by Grégoire and Langlais (2018), despite the circumstance that it was never explicitly trained on a labeled version of the pseudo-comparable corpus —as opposed to their classifier— and also despite the fact that it faces a much higher concentration of mal-aligned sentence pairs as it goes

through all source-target combinations. While the setup between the two approaches is not completely the same due to the slight differences in the pseudo-comparable corpus building, this shows that $\text{Transformer}_{\text{margP}}$ is —at least on this rather synthetic corpus— capable of extracting with an accuracy and precision that is competitive to systems which were trained in a supervised fashion.

4.3 Experiments II: Semi-Supervised Learning

In the previous experiments, word embeddings pre-trained on large monolingual corpora were used to initialize the system. However, while there are scenarios where bilingual data is sparse but monolingual data plentiful, other low-resource scenarios are sparse in both monolingual and bilingual settings. In such a case, training word embeddings on several million sentences is not always feasible. However, if a small parallel corpus is available, pre-training the NMT model on it in order to obtain a first initialization of the sentence representations might be a possible alternative.

This approach can also be of interest when a trained NMT system is already available but should be further enhanced with additional monolingual data. One concrete example would be domain adaptation.

To explore this semi-supervised scenario, we pre-train bilingual English-French models on $500k$ parallel sentences, followed by the extraction of additional data from Wikipedia.

4.3.1 Model Specifications

Experiments with both LSTM and transformer architectures are performed. For both systems, a bilingual *base* model is trained on $500k$ parallel sentences from the English and French Europarl corpora (EP base). On top of these, the linked WP articles (WP Comparable $en-fr$) are used to extract parallel sentences to train on. This results in a total of 4 models with the following specifications. All LSTM and transformer model share the same general specifications of their counterparts in 4.2.1.

- **LSTM**$_{\text{base}}$: Trained for 10 epochs on $500k$ sentences from EP base.
- **LSTM**$_{\text{margP+WP}}$: Built on top of LSTM$_{\text{base}}$ and extracts from en-fr WP comparable. As LSTM$_{\text{margP}}$ (4.2.1), it uses both representations C_h and C_e in the low-permissibility mode. The margin-based scoring function margin$(S_{\text{L1}}, S_{\text{L2}})$ is used.
- **Transformer**: Analogous to the LSTM models, a Transformer$_{\text{base}}$ model and Transformer$_{\text{margP+WP}}$ are trained with the same settings.

4.3.2 Results and Discussion

The translation performance of the base models and the semi-supervised models is shown in table 4.11. These are compared with Grégoire and Langlais (2018) on $en2fr$ due to their very similar setup. Their experiments, too, use $500k$ parallel EP sentences to train

Reference	Corpus, $en + fr$ sent. (in millions)	BLEU nt13 $en2fr$	BLEU nt13 $fr2en$	BLEU nt14 $en2fr$	BLEU nt14 $fr2en$
Semi-Supervised NMT					
Grégoire and Langlais (2018)	EP (0.5)	17.63	–	–	–
Grégoire and Langlais (2018)	+ WP (1.5)	27.10	–	–	–
Experiments II					
$\text{LSTM}_{\text{base}}$	EP (0.5)	18.44	16.2	19.18	16.37
$\text{LSTM}_{\text{margP+WP}}$	+ WP (12+8)	20.91	19.78	22.04	20.52
$\text{Transformer}_{\text{base}}$	EP (0.5)	17.46	15.92	17.78	15.81
$\text{Transformer}_{\text{margP+WP}}$	+ WP (12+8)	24.88	24.15	27.57	26.55

Table 4.11: BLEU scores achieved by the *en-fr* semi-supervised models on *newstest2013* (nt13) and *newstest2014* (nt14) as calculated with `multi-bleu.perl`. Corpus sizes are given in millions, where the size of WP for Grégoire and Langlais (2018) is the final number of pairs selected from WP, while for our experiments it constitutes the size of the comparable corpus extracted from.

an initial model on top of which they add up to $1.5M$ extracted sentences. Their major difference from our approach is that they train a separate classifier to assign a similarity score to sentence pairs. They then take the $1.5M$ *top scored* pairs to continue training their NMT system. In our case, this extraction and training happens simultaneously, which means that especially at the beginning of training it will not extract all optimal candidates. Nevertheless, the systems use the same data and it is therefore interesting to see a comparison.

The method of using separate extraction as in Grégoire and Langlais (2018) outperforms the margP+WP models. This is on one side because of their pre-selection of the top n most similar sentence pairs for training. However, in the following section, some of the difficulties that our semi-supervised models face, and which lead to these results, will be examined and discussed.

Focusing on the base models, it is surprising to see that the LSTM base model outperforms its transformer counter part. This may be due to the small amount of data used and that these do not suffice to properly optimize the larger amount of parameters in the transformer to benefit from its usually superior representations and translation quality. After joint extraction and training, the transformer again outperforms the LSTM by about 4 BLEU points.

In the following sections, the details of the extraction and the representation will be discussed in order to gain a deeper understanding of the benefits and difficulties of both models.

	Unique Accepted Pairs
$\text{Transformer}_{\text{margP+WP}}$	1.792.297
$\text{LSTM}_{\text{margP+WP}}$	478.612

Table 4.12: Unique pairs accepted from *en-fr* WP during the training of both semi-supervised models.

4.3.2.1 Unique Accepted Sentence Pairs and Similarity Distributions

Similarly to the analysis done with the unsupervised models, we will now focus on the extracted pairs and their scores.

As can be seen in 4.12, $\text{Transformer}_{\text{margP+WP}}$ extracts more than three times as many pairs than its LSTM counterpart. The dimensions of the **difference in extraction** can be seen more clearly when observing how the unique pairs are accumulated over the epochs by both systems in figure 4.9. As opposed to the unsupervised margP systems, the sudden increase in newly accepted pairs in epoch 2 is less pronounced. However, in $\text{Transformer}_{\text{margP+WP}}$, the first epoch on the new data set is still of importance for adapting the representations, as it is still followed by a major increase in extracted data in epoch 2. Nevertheless, the total increase in extracted data from epoch 1 until the end of training is still far from what has been observed in the unsupervised $\text{Transformer}_{\text{margP}}$, which shows a 10-fold increase in accepted pairs before reaching convergence. The approx. 88% increase of extracted pairs between epoch 1 and 6 in $\text{Transformer}_{\text{margP+WP}}$ seems surprisingly low compared to that. For $\text{LSTM}_{\text{margP+WP}}$, this effect is even more pronounced, as it stays around the $400k$ mark throughout the whole training process.

When taking into account the **difference in average scores** between the rejected and accepted pairs, one can see that they already have a large divergence. As both systems have already been pre-trained on $500k$ EP sentences, they start their training with C_e *and* C_h representations that have been adapted to the pre-training corpus. This is quite different from the situation of the unsupervised models in experiments 1, which only had properly initialized C_e representations while C_h was initialized randomly. Because C_e and C_h already have been adapted to the pre-training corpus, the discrepancy in average similarity score between accepted and rejected pairs is already high at the beginning of training[13]. This gap between the two distributions —while being a positive sign of representation learning for the unsupervised models in experiments 1— in fact poses a problem for the semi-supervised models. Since the representations have been adapted to the pre-training corpus, identifying new parallel sentences in the new corpus —which may vary in style and domain— becomes difficult out of two major reasons. Firstly, some words are new or rare and might be overshadowed by the embeddings of

[13]Approximately 0.66 scoring points for both semi-supervised models vs. approx. 0.05 for most unsupervised models in experiments 1

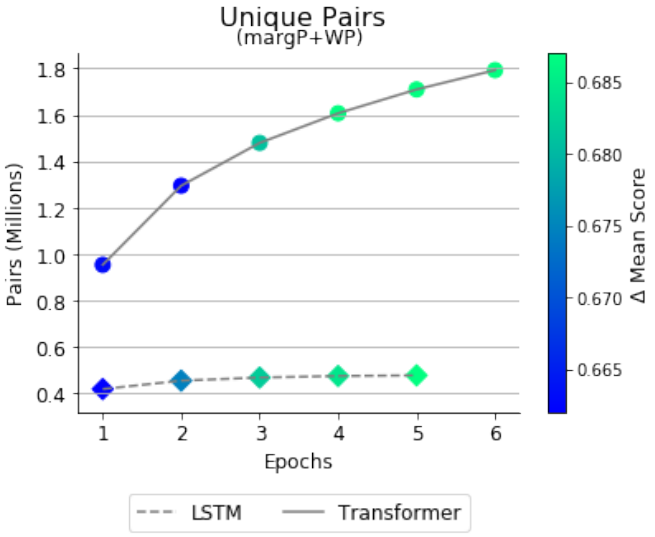
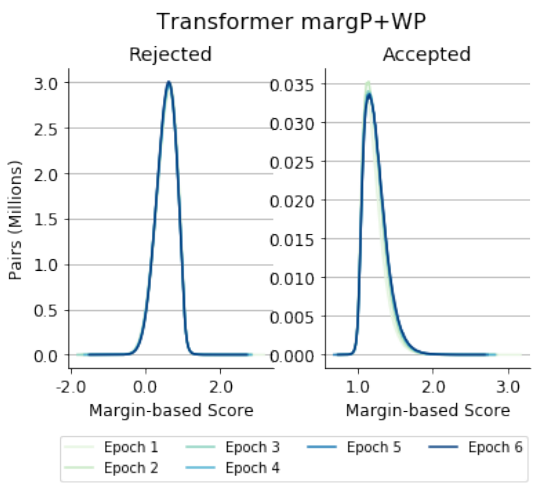


Figure 4.9: The number of unique pairs extracted by both *en-fr* semi-supervised margP+WP models as training progresses. The color map indicates the difference in the average score of accepted and rejected pairs.

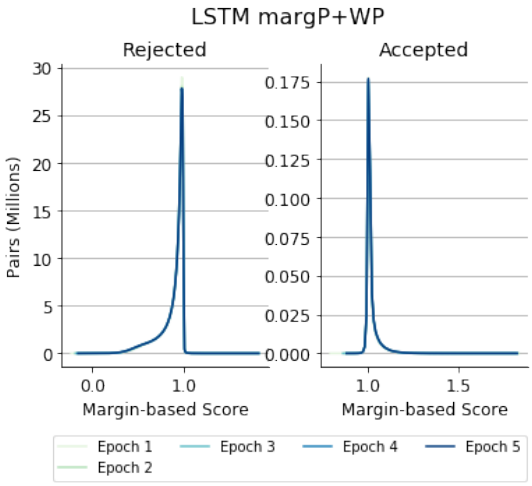
well-known words which might contain larger values than the randomly initialized embedding of a rare word. Secondly, the words in the new domain carry different nuances, meanings or usages as from the original domain, making it hard to identify their correct counterpart in the target representations. Or put differently: As there already is a large gap between the two distributions, it is harder for sentence pairs in the rejected distribution to pass to the cluster of accepted pairs.

This change in the two distributions is rather stagnant, as compared to its unsupervised counterparts. This becomes even more clear when examining the **distribution of mean scores** of accepted and rejected pairs over the epochs in figures 4.10a and 4.10b. Only minimal change in the distributions can be observed. This is much different from $\text{Transformer}_{\text{margP}}$, and most other unsupervised Transformer models, which did show how the two distributions are pushed apart. As these are already separated in our semi-supervised models, the change, however, stays small.

It can therefore be said, that the pre-training on a parallel data set was not necessarily beneficial for extraction. As the representations are already adapted to the pre-training corpus and the average scores of the accepted and rejected distributions already quite separated, it is harder for new sentences to be accepted. Also, as the representations already contain larger values from pre-training, it takes longer for the training steps on the new data to alter them.



(a) Similarity distribution (Transformer)



(b) Similarity distribution (LSTM)

Figure 4.10: Distribution of margin-based score $\text{margin}(S_{\text{L1}}, S_{\text{L2}})$ for rejected and accepted pairs when extracting from *en-fr* Wikipedia using both margP+WP models.

4.3.2.2 Intersections: Direction vs. Representation

In the previous section we have observed that pre-training the models representations on a parallel corpus is not necessarily beneficial for identifying and accepting pairs. Now, a special focus will be laid on the rejected sentence combinations.

In figure 4.11, the top graph shows the percentage of pairs rejected by the primary filter, while at the bottom we can see the percentage of sentence pairs rejected by the representations. For both $\text{Transformer}_{\text{margP+WP}}$ and $\text{LSTM}_{\text{margP+WP}}$, the quota for the secondary filter is quite alike their margP counterparts with approximately 0.2% and 1.5% respectively. In general it can be said that both filters do not change dramatically in relevance as the epochs progress.

Interestingly, the curves for the LSTM and transformer secondary filters do not progress similarly. The LSTM's secondary filter gains in relevance, as is expected from what we have previously observed in its unsupervised counterpart $\text{LSTM}_{\text{margP}}$. That is, more pairs are accepted by the primary filter and then rejected by the representation intersection. We can deduce that for C_h, which is chosen for the visualization of the primary filter, the representations are adapted quickly to the new data, allowing them to pass. C_e, however, does not change as quickly, leading to a smaller intersection between the two representation types and thus an enlarged relevance of the secondary filter for rejection.

On the other side, $\text{Transformer}_{\text{margP+WP}}$ decreases the relevance of its secondary filter by the second epoch. That is, less sentences pass the primary filter to get the

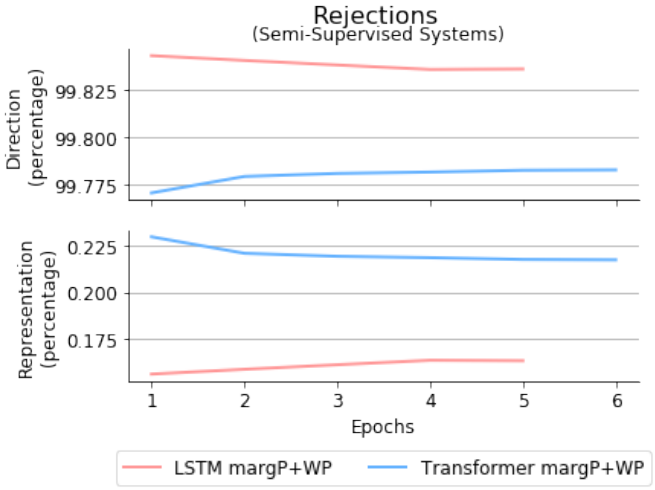


Figure 4.11: Percentage of sentence pairs rejected by both semi-supervised *en-fr* systems via the source-target direction intersection (top) and the representation intersection (bottom).

chance to be rejected by the secondary filter. However, most sentences that *do* pass the primary filter also pass the secondary filter, which is reflected in the larger amount of extracted sentences for $\text{Transformer}_{\text{margP+WP}}$ as opposed to its LSTM counter part. Therefore, the shrinking relevance of the secondary filter might just be a reflection of the enlarged intersection between both representation types, meaning that both C_h and C_e have adapted enough to the data by epoch 2 to accept and reject the same type of sentence combinations.

4.3.3 Control Experiments: Extraction Accuracy

Having analyzed the translation quality, the representations and the filters, we now want to focus on the extraction accuracy. Analogous to the control experiments for the unsupervised models, we cannot analyze the extraction performance directly on the semi-supervised models, since there is no underlying ground truth of parallel sentences available for En-Fr. Therefore, after training the transformer and LSTM on the same $500k$ EP parallel corpus, they extract and train from a **pseudo-comparable corpus**.

4.3.3.1 Design

The pseudo-comparable corpus is created with very similar statistics as the one used for the unsupervised control experiments. However, as we have to exclude all sentences from EP that are part of the parallel base corpus, the total size of the pseudo-comparable corpus is halved. Nevertheless, the general statistics of the corpus are the same as in the

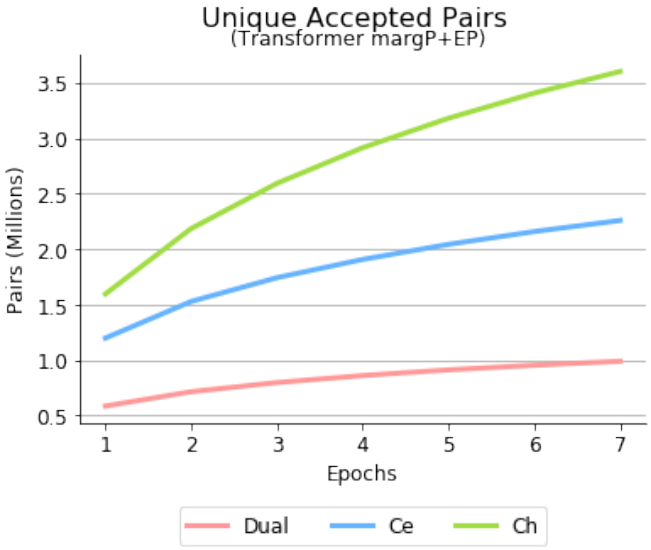


Figure 4.12: Number of unique sentences accepted by the secondary filter (dual) and the single representations C_e and C_h of Transformer$_{\text{margp+EP}}$ used for the *en-fr* semi-supervised control experiment.

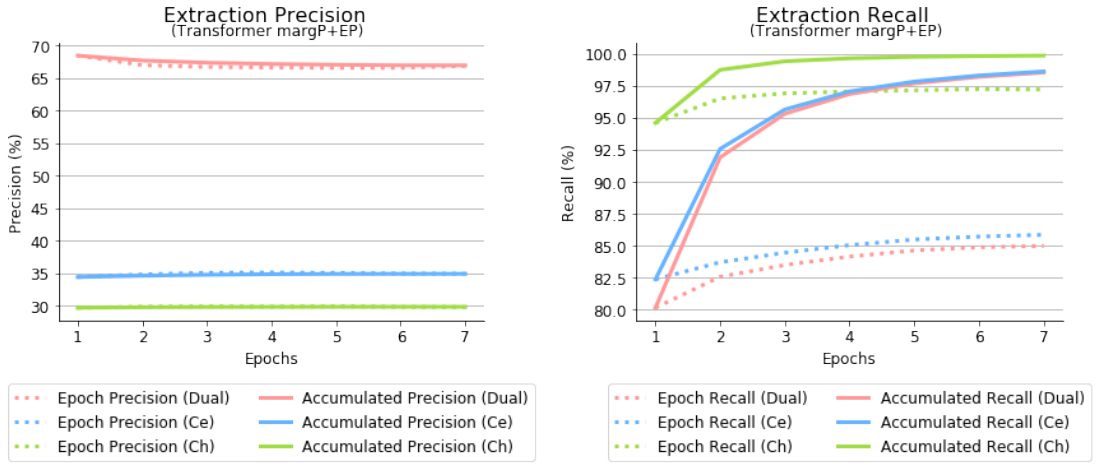
unsupervised control experiments. That is, we use an initial noise ratio of 1:4 parallel-negative samples. As the number of true parallel sentences here is $0.5M$, the number of negative samples is thus $2.5M$. We then add additional sentences without translations to the English articles to reach the mean English-French article length of 70 and 46 respectively.

In analogy to the unsupervised control experiments, we take the best performing model —Transformer$_{\text{margP}}$— to perform the extraction and training from the pseudo-comparable corpus, under the name of Transformer$_{\text{margP+EP}}$.

4.3.3.2 Results

When observing the number of **unique accepted pairs** from the pseudo-comparable corpus in figure 4.12, we can see that its curves are very different from the unsupervised control experiment. Firstly, due to the pre-training on the base corpus, C_h is already initialized on actual data and therefore already starts its extraction with a higher rate of accepted pairs than C_e. Further, the *turning point* at epoch 2 —where we can see the steepest increase in accepted pairs— is very little pronounced. Instead, both representations gradually find more sentences as training progresses, very much like in the main semi-supervised experiment on WP.

As C_h extracts more unique pairs as training progresses, its **recall** grows to 99.9%, as can be seen in figure 4.13b. Also the final extraction (dual) reaches high levels of



(a) Precision Transformer$_{\text{margP+EP}}$ (b) Recall Transformer$_{\text{margP+EP}}$

Figure 4.13: Precision and recall of semi-supervised Transformer$_{\text{margP+EP}}$ on the *en-fr* pseudo comparable EP corpus.

recall, mostly bounded by the less permissive C_e. Especially the second epoch sees a major increase of more than +10% for both C_e and the secondary filter. This correlates with the steeper increase in accepted unique pairs by epoch 2.

While more pairs are accepted and the recall increases, the final extraction **precision** decays slightly as training progresses. This is surprising, as it is the opposite effect from what was observed in the unsupervised control experiments with Transformer$_{\text{margP}}$, where the precision increased throughout the training. Further, the primary filters C_e and C_h do not decay in precision, and instead increase slightly[14]. That is, due to the increased number of sentences accepted by the primary filters without a significant increase in precision, the number of false positives is also raised. This higher number of false positives then leads to a higher probability of the same negative pairs having been accepted by both primary filters, and thus passing the secondary filter. In other words, as training progresses in this semi-supervised setting, the more similar mistakes C_e and C_h make, leading to a decaying final extraction precision of the model. One reason for this may be the fact that both representations have gone through the same pre-training on EP base, and thus have more similar biases and tendencies to make wrong classification decisions.

This lower extraction precision is also reflected in the main experiment's results by the much lower BLEU score and much smaller amount of unique accepted pairs due the internal representations being damaged by the larger influx of false positives.

[14]We see an increase of +0.2% in C_h from the beginning of the extraction until the end. For C_e, this increase is +0.5%.

4.3.4 Development Experiments: Naive Online Extraction

Now that we have analyzed various components of our online extraction systems both in the unsupervised and the semi-supervised scenario, we will make a quick detour into one of the earlier extraction systems developed for this project in order to underline the main achievements that our final margP model brought.

At the beginning of the project, the ideas of using two types of representations, filtering via *intersections of intersections* and using a margin-based scoring function instead of cosine similarity directly were not yet developed. Instead, the naive idea to online parallel sentence extraction was to take a simple encoder-decoder and then:

1. Retrieve the hidden state representations C_h of a source and a target candidate.
2. Calculate their cosine similarity.
3. If the cosine similarity surpasses a manually set threshold, accept the pair.
4. If enough pairs are collected to create a batch, train.
5. Continue extraction and training in a loop.

Taking a quick look at these early experiments shows how difficult the task of online parallel sentence extraction is and how much the system has developed throughout this project.

The initial idea was to train the system in a **semi-supervised** fashion, very similarly to experiments 2, where a $500k$ EP parallel corpus is used to pre-train the model before extracting and training on WP. However, in order to be able to measure the extraction performance as well, initial experiments were performed extracting from the $1.4M$ EP pseudo-comparable corpus with 50% negative samples.

These initial experiments were mostly concerned with the **threshold**, that needed to be set manually. This was done by examining the cosine similarity distribution of sentence combinations in the corpus, and defining the top p percentile to accept. The threshold was then automatically set, such that only the chosen percentile will pass. However, as the representations change —and no margin-based scoring function or filtering was used— the initial threshold would quickly be overcome by unwanted candidate pairs. Therefore, one technique was to use a **dynamic** threshold, which would change based on a set schedule (of n training steps), either decaying or increasing the threshold. In this case, the percentile is increased or decreased by 1 at each epoch.

In table 4.13, the *averaged* BLEU score for both language directions $en2fr$ and $fr2en$ can be observed with varying threshold types. There are two major points of

Threshold Type	Initial Threshold	Averaged Bleu
Static	0.079 ($p = 50$)	18.59
	0.241 ($p = 70$)	20.62
	0.274 ($p = 80$)	21.09
	0.317 ($p = 90$)	21.25
Growing	0.079 ($p = 50$)	19.7
	0.271 ($p = 80$)	21.48
Decaying	0.077 ($p = 50$)	18.5
	0.271 ($p = 80$)	20.68

Table 4.13: Naive online extraction: Averaged BLEU score on *newstest2013* for $en2fr$ and $fr2en$ for varying extraction thresholds (cosine similarity). p is the percentile chosen for defining the cosine similarity-based threshold.

interest when observing these results. Firstly, all of the cosine similarities used for the threshold are surprisingly low, compared to what has been observed to be the mean cosine similarity of accepted sentences by simP, which circulates around the high 0.9's (see figure 4.3d for comparison). However, when we take into account that all models in the main experiments 1 and 2 reject around 99.98% of all sentence combinations, it is clear how a seemingly high percentile such as 80 or 90 for setting the threshold is still not sufficient for filtering out unwanted pairs.

Secondly, while in general we can observe that higher thresholds perform better, the system with the growing threshold and $p = 80$ outperforms the system with a static threshold of $p = 90$. As mentioned before, the representations change over time, and as has been analogously observed in the similarity distribution of simP (figure 4.3d), false positives push all sentences to be more similar. To overcome this, one would optimally avoid to accept false positives from corrupting the representations. This has been the case for many systems in our main experiments 1 and 2, where, as training progresses, the accepted and rejected distributions are pushed apart instead of moving closer. However, as the distribution of average similarity *is* moving and this is not a threshold-less approach, the need for a growing threshold arises to adapt to the growing cosine similarities. Nevertheless, this is not easy to define and leads to complex scheduling systems, that may be based on linear growth, exponential growth or even a scheduler based on the systems perplexity or cross-entropy. This need for threshold scheduling is one of the major flaws of naive online parallel sentence extraction.

Lastly, we want to look at the naive approach and see which problems have been solved by our final online extraction system:

1. **Scalability of cosine similarity**: The naive approach (and simP) use cosine similarity as their scoring functions. As discussed before, it leads to problems

of scale between different sentences, making it less suitable for identifying good sentence pairs. The margin-based scoring function $\text{margin}(S_{\text{L1}}, S_{\text{L2}})$ solved this.

2. **Parameter exploration**: Even with a seemingly high percentile of 90 to set the threshold, many sentence combinations are false positives. Setting this value appropriately is not trivial and needs to be explored for each and every data set we use. This tuning is costly in time and resources, and using our combination of directional primary filters and the intersection of two representation types in the secondary filter is the key to threshold-less extraction and training.

3. **Threshold schedules:** Not only is it time-consuming to find an initial threshold, but adapting it appropriately to the changing representations during training is complex. As our current system's filtering is based on these changing representations, they can adapt their extraction decisions to the current state of the system. This makes the threshold-less approach also stable throughout the training process.

4.4 Applications

4.4.1 Experiments III: Low-Resource NMT

One of the major application of our proposed self-supervised NMT system is low-resource NMT. The system can be trained without any parallel data, which is a powerful feature for training an NMT system on an under-resourced or zero-resourced language pair. Yet, as the unsupervised experiment in 4.2 focus on exploring the different properties of the technique itself, those experiments were performed on an actual high-resource language pair in order to stay comparable to other work. In this section, however, the focus lies on using our technique for an actual **low-resource pair**, namely Gujarati-English. One major reason for choosing this language pair —besides being truly low-resource— is the existence of a WP parallel corpus that was created for WMT 2019. This will allow us to calculate the extraction accuracy of the system on WP data, instead of resorting to pseudo-comparable corpora as has been done in previous control experiments.

4.4.1.1 Data

We pre-train **word embeddings** on monolingual data. For English, the *en* WP Edition (see table 4.2), as well as NewsDocs (table 4.3) and the English versions of the crawled corpora *News18* and *Zeenews* (4.4) are used. Gujarati embeddings were trained on the concatenation of all crawled Gujarati articles (*Divya*, *News18*, *Gujarat Samachar*, *Sandesh* and *Zeenews*), the *Bible* corpus, *WMT19 Localisation*, *WMT19 Crawl*, *NewsCrawl*, *CommonCrawl* and the Gujarati monolingual Wikipedia. This results in monolingual corpora sizes of about $52M$ and $6.5M$ for English and Gujarati respectively.

We use *newsdev2019 dev* for development and both *newsdev2019 test* as well as *newstest2019* for testing.

We create a parallel corpus of the shuffled concatenation of all Gujarati-English corpora listed in table 4.1 — which are not included in the development and test sets— totaling a size of approximately $200k$ pairs. A base model is trained on this data.

4.4.1.2 Model Specifications

The same model specifications as $\text{Transformer}_{\text{margP}}$ are used. Three models are trained:

- **Transformer**$_{\text{base}}$: Model trained on the concatenation of the above-mentioned parallel data.

	BLEU (*newsdev2019 test*)		BLEU (*newstest2019*)	
	en2gu	*gu2en*	*en2gu*	*gu2en*
$\text{Transformer}_{\text{base}}$	3.73 (1.58)	6.44 (3.52)	0.79 (0.31)	6.44 (5.21)
$\text{Transformer}_{\text{margP+WP}}$	2.19 (0.68)	4.55 (2.58)	0.31 (0.09)	4.86 (3.69)
$\text{Transformer}_{\text{+extracted}}$	3.58 (1.47)	5.87 (3.45)	0.7 (0.28)	6.34 (5.08)

Table 4.14: Results of the *en-gu* models on the self-defined test set and *newstest2019*. Tokenized BLEU on *en2gu* is calculated on the romanized and tokenized versions of the Gujarati corpus, while untokenized BLEU —in brackets— on the original untokenized version written in Gujarati script. $\text{Transformer}_{\text{margP}}$ has a BLEU of 0 in all categories and is not included here.

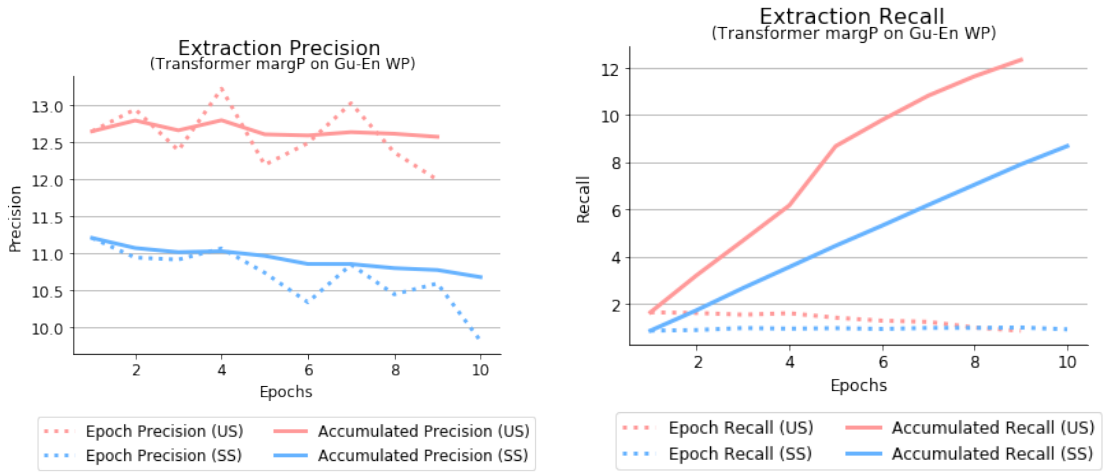
- **Transformer**$_{\text{margP+WP}}$: Takes $\text{Transformer}_{\text{base}}$ for initialization and extracts from the *en-gu* comparable WP.
- **Transformer**$_{\text{margP}}$: Model using pre-trained embeddings only to extract from *en-gu* comparable Wikipedia.
- **Transformer**$_{\text{+extracted}}$: Trains on the base parallel corpus and the extracted pairs of the last epoch of $\text{Transformer}_{\text{margP}}$.

4.4.1.3 Results

In table 4.14 the BLEU scores obtained on both the smaller self-defined test set as well as the official *newstest2019* corpus are reported. As the unsupervised model $\text{Transformer}_{\text{margP}}$ has a BLEU of 0, it is not contained in the table. It becomes clear that all models have a relatively low performance, which is due to the small amount of data. What calls for attention, however, is the fact that the models that extract additional data from WP decay the translation performance of the base model. This effect is slightly reduced when the additional data is extracted in an unsupervised fashion with $\text{Transformer}_{\text{margP}}$ and then added to the base-model afterwards as in $\text{Transformer}_{\text{extracted}}$. The reasons for this will be discussed in more detail in the following by taking a closer look at the extraction precision and recall.

In figure 4.14 we can observe the precision and recall extracting from the *en-gu* WP articles using the semi-supervised model $\text{Transformer}_{\text{margP+WP}}$ and the unsupervised $\text{Transformer}_{\text{margP}}$. However, these results should be taken with a grain of salt, simply because the *reference* we compare with was extracted automatically[15] and therefore many sentences that may actually be parallel are not included. As such, the precision values reported here are not necessarily good indicators of the true precision. How-

[15]See `http://www.statmt.org/wmt19/translation-task.html`



(a) Precision ($en - gu$)

(b) Recall ($en - gu$)

Figure 4.14: Accumulated and epoch-specific precision and recall of the unsupervised (US) and semi-supervised (SS) *en-gu* models $\text{Transformer}_{\text{margP}}$ and $\text{Transformer}_{\text{margP+WP}}$ respectively.

ever, the recall values are of interest, as they indicate how many parallel sentences of a representative subset have been identified and extracted.

When comparing the **recall** scores, it becomes clear that $\text{Transformer}_{\text{margP}}$ covers more pairs of the parallel subset than its semi-supervised counterpart. This is surprising given the much smaller amount of data used for training the Gujarati embeddings and their resulting less-than-optimal initialization. However, this reflects the observations in experiments 2, stating that initialization via pre-trained embeddings leads to better extraction than pre-training the model on a small parallel corpus.

The higher recall of the unsupervised model is also reflected in the larger amount of **unique extracted pairs**, as can be observed in 4.15. While the semi-supervised model levels out at around $3k$ pairs, the unsupervised model extracts more than twice as many.

Nevertheless, general recall is very low as compared to $\text{Transformer}_{\text{margP}}$ on *en-fr* WP, where it reached scores of 0.95 and above. The characteristic sudden increase in recall and extracted pairs, which in earlier described unsupervised models indicated a *turning point* or *domino effect* in training where both representations have adapted sufficiently to the data set to extract increasingly more (high-quality) data, is not found here. The reasons for this are likely two-fold. Firstly, the corpus is too small to collect enough representative data from, such that the representations are over-fitted on a small and overly specific subset. They thus never adapt to the true distribution of the corpus. This is further enforced by the rather meager pre-trained Gujarati embeddings, which do help the model to identify a small subset of parallel data, but not enough to push the model to extract enough sentences from the corpus to reach the above-mentioned

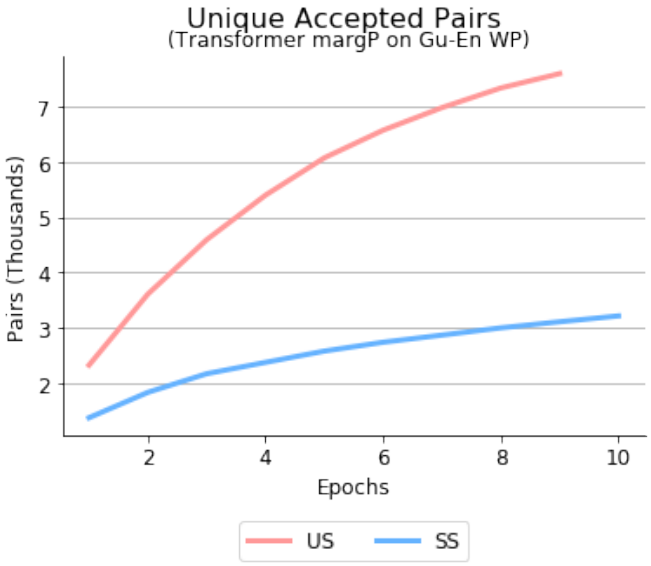


Figure 4.15: Number of unique extracted pairs from *en-gu* Wikipedia by semi-supervised Transformer$_{\text{margP+WP}}$ (SS) and unsupervised Transformer$_{\text{margP}}$ (US).

turning point. Nevertheless, neither the rather high-resource *en-fr* word embeddings have pushed the margP models to this turning point. Most margP models made this leap by the second or third epoch, simply by extracting enough data from the corpus to adapt to the general corpus. Therefore, the lack of comparable data is the major reason for the low recall.

This being the recall, we also have indications of a relatively low **precision**, apart from what has been deduced from the comparison with the reference. The decayed BLEU score when adding WP data to the base model indicated that both semi-supervised and unsupervised approaches do not reach sufficient precision to overcome the negative effects of the extracted false positives. However, adding the extracted data from the last epoch of the unsupervised model on top of the base model, as has been done in Transformer$_{\text{+extracted}}$, this effect is reduced, indicating that the quality of the extracted data from the unsupervised model is of higher precision than that of its semi-supervised counter part.

All in all, it can be said that the method is dependent on sufficiently large comparable corpora in order to reach well-adapted representations for extraction. Also, higher quality embeddings trained on larger amounts of monolingual data are beneficial for an increased precision. Extracting using the unsupervised method on pre-trained embeddings and then using that extracted data on top of (or mixed with) a base corpus for training is advised.

In order to make this technique suitable for very low-resource scenarios such as *en-gu* translation, it is necessary to explore representation learning for low-resource languages

as well as multi-task representation learning. In our case, adding mode languages to the multilingual setting, such that the model is trained additionally on several similar languages such as Hindi, Bengali etc. could help improve the internal representations and therefore the final extraction.

4.4.2 Experiments IV: Corpus Cleaning

The web is filled with multilingual content, and crawlers that specialize on extracting bitexts[16] from it make it easy to identify and extract parallel data that can be used for training MT systems. However, crawled data tends to be noisy and sometimes filled with redundant or useless sentences. Corpus cleaning which goes beyond running Moses scripts and hand-written heuristics is therefore of interest. We want to use our joint training and extraction method on a crawled parallel corpus to observe whether it is able to identify useful sentences to extract and train on.

For these purposes, we initialize the models embeddings on the English and German version of *NewsDocs* and then start extracting sentences from *en − de ParaCrawl* (`BiCleaner v.3.0`). To compare how allowing the system to extract sentences from the corpus differs from simply training on the crawled corpus, a reference model is trained directly on *ParaCrawl*.

4.4.2.1 Model Specifications

We initialize $\text{Transformer}_{\text{margP}}$ (see 4.2.1) on the word embeddings trained on NewsDocs. These two models are then used to extract and train from *ParaCrawl*. For the reference model, $\text{Transformer}_{\text{base}}$, a transformer model with the standard specifications in 4.2.1 is trained directly on the *ParaCrawl* corpus without filtering.

4.4.2.2 Discussion and Results

The BLEU scores on *newstest2014* by both the base model and our filtering model can be seen in table 4.15. We also add results achieved by Artetxe and Schwenk (2018) in a similar setup. However, their results are not comparable to ours, since they used a different version of the corpus (`BiCleaner v.1.2` for their base, and raw ParaCrawl for filtering), performed various filtering steps based on heuristics before-hand and trained on one language direction only. Nevertheless, their results should be mentioned as their general approach of using margin-based scoring for filtering is not dissimilar from ours. In fact, their filtering method improves on their baseline, which is also the case in our

[16] For example `https://github.com/bitextor/bitextor/tree/v7`

	Unique Pairs	*en*2*de*	*de*2*en*
Artetxe and Schwenk (2018) Base	17.4M	30.05 (29.37)	–
Artetxe and Schwenk (2018) Filtering	10.0M	31.19 (30.53)	–
$\text{Transformer}_{\text{base}}$	31.1M	15.4 (11.35)	19.01 (15.33)
$\text{Transformer}_{\text{margP}}$	16.7M	18.68 (14.03)	22.35 (18.2)

Table 4.15: Tokenized BLEU as well as untokenized BLEU (in brackets) of the filtering models on *en-de newstest2014*.

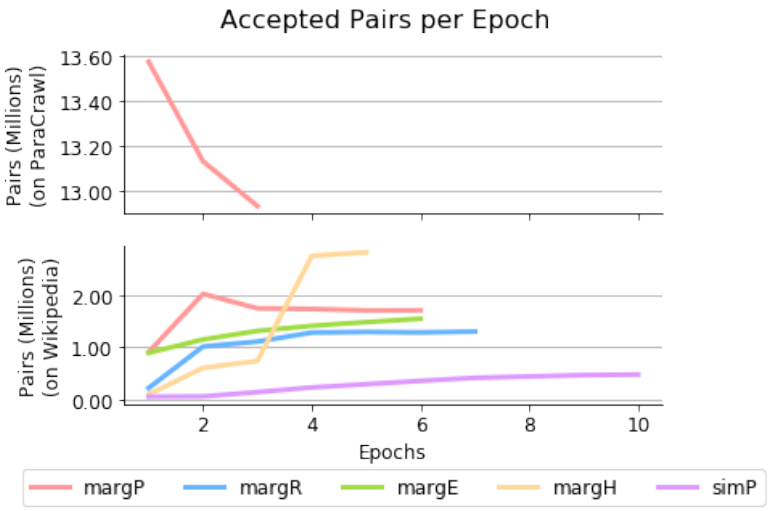


Figure 4.16: Number of pairs accepted at each epoch on $en-de$ ParaCrawl (top) and $en-fr$ Wikipedia (bottom) using various models.

setting.

We gain more than 3 BLEU points when using $\text{Transformer}_{\text{margP}}$ when filtering the pre-filtered ParaCrawl corpus. This suggests that it further improved the filtering of a corpus that was already pre-filtered by another algorithm, removing more noisy or simply less useful sentences. We can observe this gradual reduction of the corpus when looking at the number of non-unique pairs extracted at each epoch.

When we compare the **number of non-unique pairs** accepted by $\text{Transformer}_{\text{margP}}$ on ParaCrawl (top in figure 4.16) and most other models on Wikipedia (bottom), the pattern is quite different. While almost all models generally increase the number of pairs accepted in each epoch throughout the training, this is not the case for the filtering model. In fact, this is what is expected from a model that —as the representations are adapted to the corpus— filters with increasing rigidity for useful sentences. However, the curves are not completely comparable. On WP, most models used the first epoch to adapt their representations (*initialization*), and in the second or third epoch started

extracting significantly more (*turning point*). By the end of training, the number of pairs extracted per epoch dropped slightly, which, according to the control experiments in 4.2.3 most likely comes with a last increase in precision before converging. When extracting from the much larger ParaCrawl, however, the epochs are longer, and therefore the initialization phase and the turning point already appear in the first epoch. As such, no increase in accepted sentences from epoch 1 to epoch 2 is observable. Likely, the drop in extracted sentences is similar to the one observed in the later epochs of $\text{Transformer}_{\text{margP}}$ when extracting from WP. Meaning, that the *precision*[17] is rising, which in turn pushes down the number of accepted pairs. That this reduction of the corpus was in fact based on positive decisions is reflected in the significant increase in BLEU as described above.

[17] *Precision* may not be the most suitable term here. According to the ParaCrawl base-line, each originally mapped sentence pair in the corpus should be a true positive. However, not all of them might be useful for training.

Chapter 5

Conclusion and Future Research

In the scope of this project, we have developed a joint architecture to extract data and train NMT systems simultaneously using the emerging NMT systems internal representations to select the data. This is a form of self-supervision alternating between two tasks that support each other in an incremental fashion.

A special focus was laid on studying the workings of the sentence representations and how these can be exploited to provide an adequate function for the selection and filtering process, without the need of an additional hyperparameter that depends on the input corpus.

The final system comes with three key improvements over the initial vanilla extraction system: Firstly, The usage of an **iterative and joint extraction and training** system, which is especially beneficial for unsupervised scenarios where the amount and quality of extracted data increases as training progresses since the system is able to exploit earlier extraction decisions and their impact on the NMT internal sentence representations. Further, the usage of the **margin-based scoring** function (Artetxe and Schwenk, 2018) to score source-target candidates in order to avoid the scaling problem of cosine similarity. And last but not least, filtering of candidates using the **intersection of two representation types** C_h and C_e which leads to high precision extraction decisions without the need of the data specific and resource consuming effort of prior threshold exploration.

The various sub-models that resulted from the development of this system were studied in detail in an unsupervised setting using both LSTM and Transformer architectures. There we have examined the roles of the representations C_e and C_h, concluding that they in fact complement each other in performing the extraction task. Not only do they make different mistakes, which are then filtered out by their intersection, but their **different pace in adaptation improves the decisions** accordingly. The interplay of the rigid

C_e enforcing high precision and flexible C_h encouraging recall has further been observed in the related control experiments. Apart from the quantitative experiments focusing on the representations and their effects on the system, we have also performed a qualitative analysis of the extraction from a Wikipedia article as training progresses. There we have seen the importance of a shared BPE encoding as well as common numerals and other relevant homographs for the systems extraction decisions, especially at the beginning of training.

The best performing system that emerged from the unsupervised experiments is $\text{Transformer}_{\text{margP}}$, which implements all of the above key features and uses strict filtering in the low permissibility mode. It outperforms other state-of-the-art unsupervised NMT and SMT-NMT systems on the same test sets while using significantly smaller amounts of training data. However, we rely on comparable data to reduce the search space and speed up extraction, while other systems rely on monolingual data for training, which may be easier to obtain in certain scenarios. One major objective of future research is thus to make the extraction more efficient on non-comparable monolingual data. One technique may be to focus not solely on whole sentences, but to also allow the **extraction of smaller segments** such as phrases. This should increase the amount of parallel data available to exploit also in monolingual corpora. Further, using similarity measures or more advanced clustering techniques over corpus sections could be useful to **pre-filter sections** of large corpora by theme, in order to reduce the search space.

In later experiments, we have applied the margP model in its Transformer and LSTM version to a **semi-supervised task**. By studying its representations and the number of unique extracted pairs over the epochs, we could see that the semi-supervised setting is less beneficial for extraction. As both C_e and C_h are already adapted to the base-model, the dynamics between the two representation types that were valuable in the unsupervised case are not the same here. Especially the quality of fast adaptation to the new training data is reduced in C_h, which generally pulls down the recall when compared to the unsupervised scenario. As such, it is currently the better choice to extract using pre-trained word embeddings only as in the unsupervised case, and then add the extracted data to the existing parallel data. However, as this is not the basic idea of joint training and extraction, more practical ways in improving the performance of our system for settings where parallel data is available should be explored in the future.

One such technique to investigate is to use the parallel corpus to train a separate Transformer model, whose embeddings (C_e) will then be extracted to start an unsupervised training and extraction on comparable data. This may be an option for scenarios where not enough monolingual data is available to train high-quality unsupervised multilingual word embeddings for initialization. However, this does not solve the problem for domain adaptation or other transfer learning tasks, where we want to keep the previous C_h in order to quickly refine the model on a new task. For these scenarios, we want

to explore the effect of different **regularization techniques**. These will allow to push the values in C_h closer to 0, potentially refraining extreme values from hindering the extraction and quick adaptation on the new data set. These future experiments should be further applied to the task of **domain adaptation**.

Further, we have used our extraction system in a truly **low-resource scenario**. There we have seen that the system does need sufficient comparable data in the unsupervised case in order to adapt the representation on the training data sufficiently to start high-recall extraction. When this is not provided, the systems extraction performance stays under its potential. For these purposes, it is important to increase the amount of data extractable on a small corpus. This should be explored as a further application of the phrase-based extraction described above. We can additionally use **back-translation** for rejected pairs to further facilitate the adaptation of the representations on the data. Further, in order to improve the initial word-embeddings in such a low-resource scenario, the above-mentioned extraction of word-embeddings from NMT models trained on bilingual data may become of help.

We also saw how our system can be used to **filter** already parallel, but noisy, corpora. We have achieved a major improvement over the baseline directly trained on the noisy corpus. This indicated that the resulting reduction of the corpus was based on effective extraction decisions that improved the general quality of the corpus. However, as the experiment setup was different to the work of others, our system should in a future experiment be applied to the raw ParaCrawl corpus in order to gain insights into the performance of our system comparable to others.

To conclude, we have developed a joint extraction and training system for NMT, which performs on state-of-the-art level for unsupervised scenarios where abundant of comparable data is available. However, it is still dependent on large amounts ($\sim 10M$) of comparable data, which is still far from what one can gather in a truly low-resource scenario. In exploring both phrase-based extraction, additional back translation as well as alternative initialization techniques, we hope to make our method available to low-resourced language pairs in the future.

List of Abbreviations

AT	Auxiliary Task
BLEU	Bilingual Evaluation Understudy
BRNN	Bidirectional Recurrent Neural Network
CNN	Convolutional Neural Network
de	German
en	English
EP	EuroParl
fr	French
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
gu	Gujarati
L1	Language 1
L2	Language 2
LM	Language Model
LSTM	Long Short-Term Memory
MT	Machine Translation
NMT	Neural Machine Translation
OOV	Out-of-vocabulary
PBSMT	Phrase-Based Statistical Machine Translation
PT	Primary Task
RBMT	Rule-Based Machine Translation
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SMT	Statistical Machine Translation
TQE	Translation Quality Estimation
WMT	Workshop on Statistical Machine Translation
WP	Wikipedia

Translations of Qualitative Examples

In the following a translation of the French sample sentences used for the qualitative analysis in experiments 1. The translations are made *independently* from the translations that can partially be found in the English version of the WP article.

(a) Slender body, compressed laterally, it ends with a spike, the mouth is slightly above.

(b) The background color of the body is light beige, reddish or light grey-green with marbling.

(c) The snout is studded with small, red and black dots.

(d) The caudal fin is characterized by two bent, red to dark lines forming a "V" as well as by a black line parallel to the upper line of the "V".

(e) The characteristic features of this species, differentiating it notably from Cephalopholis urodeta, consist in the presence of two black stains on the upper side of the caudal peduncle.

(f) It is protogynous hermaphrodite, which means that the individual is first female at the sexual maturity and then becomes male.

List of Tables

Bibliography

Abdul-Rauf, S. and Schwenk, H. (2009). On the use of comparable corpora to improve SMT performance. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 16–23, Athens, Greece. Association for Computational Linguistics.

Adafre, S. F. and de Rijke, M. (2006). Finding similar sentences across multiple languages in Wikipedia. In *Proceedings of the Workshop on NEW TEXT Wikis and blogs and other dynamic text sources*.

Antonova, A. and Misyurev, A. (2011). Building a web-based parallel corpus and filtering out machine-translated text. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*, pages 136–144, Portland, Oregon. Association for Computational Linguistics.

Artetxe, M., Labaka, G., and Agirre, E. (2017). Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada. Association for Computational Linguistics.

Artetxe, M., Labaka, G., and Agirre, E. (2018a). Unsupervised statistical machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3632–3642, Brussels, Belgium. Association for Computational Linguistics.

Artetxe, M., Labaka, G., Agirre, E., and Cho, K. (2018b). Unsupervised neural machine translation. *Proceedings of the Sixth International Conference on Learning Representations, ICLR*.

Artetxe, M. and Schwenk, H. (2018). Margin-based parallel corpus mining with multilingual sentence embeddings. *arXiv preprint arXiv:1811.01136*.

Axelrod, A., He, X., and Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Barrón-Cedeño, A., España-Bonet, C., Boldoba, J., and Màrquez, L. (2015). A factory of comparable corpora from wikipedia. In *Proceedings of the Eighth Workshop on Building and Using Comparable Corpora*, pages 3–13. Association for Computational Linguistics.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155.

Bouamor, H. and Sajjad, H. (2018). H2@BUCC18: Parallel Sentence Extraction from Comparable Corpora Using Multilingual Sentence Embeddings. In *11th Workshop on Building and Using Comparable Corpora*, page 43.

Britz, D., Goldie, A., Luong, M.-T., and Le, Q. (2017). Massive exploration of neural machine translation architectures. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1993). Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pages 737–744, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder–decoder approaches. *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using RNN encoder–decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Currey, A., Miceli Barone, A. V., and Heafield, K. (2017). Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156, Copenhagen, Denmark. Association for Computational Linguistics.

Daumas, M. (1965). Les machines à traduire de georges artsrouni. *Revue d'histoire des sciences*, 18(3):283–302.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., and Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland. Association for Computational Linguistics.

Doetsch, P., Golik, P., and Ney, H. (2017). A comprehensive study of batch construction strategies for recurrent neural networks in mxnet. *CoRR*, abs/1705.02414.

Dong, D., Wu, H., He, W., Yu, D., and Wang, H. (2015). Multi-task learning for multiple language translation. 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 1723–1732, Beijing, China. Association for Computational Linguistics.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3):195–225.

España-Bonet, C. and Barrón-Cedeño, A. (2017). Lump at SemEval-2017 Task 1: Towards an Interlingua Semantic Similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 144–149, Vancouver, Canada. Association for Computational Linguistics.

España-Bonet, C., Varga, A. C., Barrón-Cedeño, A., and van Genabith, J. (2017). An Empirical Analysis of NMT-Derived Interlingual Embeddings and their Use in Parallel Sentence Identification. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1340–1350.

Firat, O., Cho, K., and Bengio, Y. (2016a). Multi-way, multilingual neural machine translation with a shared attention mechanism. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875.

Firat, O., Sankaran, B., Al-Onaizan, Y., Yarman Vural, F. T., and Cho, K. (2016b). Zero-resource translation with multi-lingual neural machine translation. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277.

Frege, G. (1884). *Die Grundlagen der Arithmetik: eine Logisch Mathematische Untersuchung über den Begriff der Zahl*. w. Koebner.

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

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence. Springer Berlin Heidelberg.

Graves, A. (2013). Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Grégoire, F. and Langlais, P. (2018). Extracting Parallel Sentences with Bidirectional Recurrent Neural Networks to Improve Machine Translation. *Proceedings of the 27th International Conference on Computation2 Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1442–1453.

Gulcehre, C., Firat, O., Xu, K., Cho, K., Barrault, L., Lin, H.-C., Bougares, F., Schwenk, H., and Bengio, Y. (2015). On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Ha, T.-L., Niehues, J., and Waibel, A. (2016). Toward multilingual neural machine translation with universal encoder and decoder. In *Proceedings of the 13th International Workshop on Spoken Language Translation (IWSLT)*.

He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T.-Y., and Ma, W.-Y. (2016). Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828.

Hochreiter, S., Bengio, Y., and Frasconi, P. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

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

Hutchins, J. (2004). Two precursors of machine translation: Artsrouni and trojanskij. *International Journal of Translation*, 16(1):11–31.

Hutchins, J. (2006). Machine translation: History. In Brown, K., editor, *Encyclopedia of Language and Linguistics (Second Edition)*, pages 375 – 383. Elsevier, Oxford, second edition edition.

Irvine, A. and Callison-Burch, C. (2016). End-to-end statistical machine translation with zero or small parallel texts. *Natural Language Engineering*, 22(4):517–548.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015). On using very large target vocabulary for neural machine translation. *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 1–10.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F. B., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2017). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association for Computational Linguist*, 5:339–351.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Khayrallah, H. and Koehn, P. (2018). On the impact of various types of noise on neural machine translation. *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 74–83.

King, G. W. and Wieselman, I. L. (1956). Stochastic methods of machine translation. *Mechanical Translation*, 3(2):38–39.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. (2017). Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72. Association for Computational Linguistics.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180. Association for Computational Linguistics.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lakew, S. M., Erofeeva, A., Negri, M., Federico, M., and Turchi, M. (2018). Transfer learning in multilingual neural machine translation with dynamic vocabulary. In *15th International Workshop on Spoken Language Translation*.

Lample, G., Conneau, A., Denoyer, L., and Ranzato, M. (2018a). Unsupervised machine translation using monolingual corpora only. *Proceedings of the Sixth International Conference on Learning Representations, ICLR*.

Lample, G., Conneau, A., Ranzato, M., Denoyer, L., and Jégou, H. (2018b). Word translation without parallel data. *International Conference on Learning Representations*.

Lample, G., Ott, M., Conneau, A., Denoyer, L., and Ranzato, M. (2018c). Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049. Association for Computational Linguistics.

Luong, T., Pham, H., and Manning, C. D. (2015a). Effective approaches to attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Luong, T., Sutskever, I., Le, Q., Vinyals, O., and Zaremba, W. (2015b). Addressing the rare word problem in neural machine translation. *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 11–19.

Maurer, A., Pontil, M., and Romera-Paredes, B. (2016). The benefit of multitask representation learning. *The Journal of Machine Learning Research*, 17(1):2853–2884.

Morishita, M., Oda, Y., Neubig, G., Yoshino, K., Sudoh, K., and Nakamura, S. (2017). An empirical study of mini-batch creation strategies for neural machine translation. *Proceedings of the First Workshop on Neural Machine Translation*, pages 61–68.

Mu, J. and Viswanath, P. (2018). All-but-the-top: Simple and effective postprocessing for word representations. *International Conference on Learning Representations*.

Munteanu, D. S. and Marcu, D. (2005). Improving machine translation performance by exploiting non-parallel corpora. *American Journal of Computational Linguistics*, 31(4):477–504.

Nisarg, J., Gupta, M., and Varma, V. (2018). Translation quality estimation for indian languages. *Proceedings of the 21st Annual Conference of the European Association for Machine Translation*, pages 159–168.

Paramita, M. L., Aker, A., Clough, P., Gaizauskas, R., Glaros, N., Mastropavlos, N., Yannoutsou, O., Ion, R., Ştefănescu, D., Ceauşu, A., Tufiş, D., and Preiss, J. (2019). *Using Comparable Corpora for Under-Resourced Areas of Machine Translation*, chapter Collecting Comparable Corpora. Springer, Cham.

Popel, M. and Bojar, O. (2018). Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1).

Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. (2007). Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International*

Conference on Machine Learning, ICML '07, pages 759–766, New York, NY, USA. ACM.

Rarrick, S., Quirk, C., and Lewis, W. (2011). Mt detection in web-scraped parallel corpora. *Proceedings of MT Summit XIII*.

Rauf, S. A. and Schwenk, H. (2011). Parallel sentence generation from comparable corpora for improved smt. *Machine translation*, 25(4):341–375.

Resnik, P. and Smith, N. A. (2003). The web as a parallel corpus. *American Journal of Computational Linguistics*, 29(3):349–380.

Schwenk, H. (2008). Investigations on Large-Scale Lightly-Supervised Training for Statistical Machine Translation. In *International Workshop on Spoken Language Translation*, pages 182–189.

Schwenk, H. (2012). Continuous space translation models for phrase-based statistical machine translation. *Proceedings of COLING 2012: Posters*, pages 1071–1080.

Schwenk, H. (2018). Filtering and mining parallel data in a joint multilingual space. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–234.

Schwenk, H., Dchelotte, D., and Gauvain, J.-L. (2006). Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 723–730. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016a). Improving neural machine translation models with monolingual data. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Sennrich, R., Haddow, B., and Birch, A. (2016b). Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, (ACL 2016), Volume 1: Long Papers*, pages 1715–1725.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:623–656.

Smith, J. R., Quirk, C., and Toutanova, K. (2010). Extracting parallel sentences from comparable corpora using document level alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 403–411, Los Angeles, California. Association for Computational Linguistics.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Son, L. H., Allauzen, A., and Yvon, F. (2012). Continuous space translation models with neural networks. In *Proceedings of the 2012 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 39–48. Association for Computational Linguistics.

Stallard, D., Devlin, J., Kayser, M., Lee, Y. K., and Barzilay, R. (2012). Unsupervised morphology rivals supervised morphology for Arabic MT. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 322–327, Jeju Island, Korea. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Utiyama, M. and Isahara, H. (2003). Reliable measures for aligning Japanese-English news articles and sentences. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 72–79, Sapporo, Japan. Association for Computational Linguistics.

Utiyama, M. and Isahara, H. (2007). A comparison of pivot methods for phrase-based statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 484–491, Rochester, New York. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Virpioja, S., Väyrynen, J. J., Creutz, M., and Sadeniemi, M. (2007). Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. *Proceedings of the Machine Translation Summit XI*, pages 491–498.

Weaver, W. (1949). Translation.

Wu, H. and Wang, H. (2007). Pivot language approach for phrase-based statistical machine translation. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 856–863.

Yang, Z., Chen, W., Wang, F., and Xu, B. (2018). Unsupervised neural machine translation with weight sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 46–55. Association for Computational Linguistics.

Zoph, B. and Knight, K. (2016). Multi-source neural translation. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34.

Zoph, B., Yuret, D., May, J., and Knight, K. (2016). Transfer learning for low-resource neural machine translation. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575.