UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

Learning Morphology for Open-Vocabulary Neural Machine Translation

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

by

Duygu Ataman

September 2019

Dedicated in loving memory to my grandmother
Mübeccel Sevim Ataman

Acknowledgements

Nothing in this dissertation would have been possible without the teachings and constant encouragement of my advisor, Marcello Federico. His strong confidence in me and all my ideas, which at times were hard to digest within the limits of rationality, was the most important support during this long journey of self discovery. I will always be truly grateful for everything I have learned from him. The first two years of my doctoral research was mainly conducted at Fondazione Bruno Kessler, where I had the privilege of working with the great researchers of the machine translation group: Roldano Cattoni, Nicola Bertoldi, Luisa Bentivogli, Mauro Cettolo, Matteo Negri and Marco Turchi, to whom I am very thankful for always making me feel home, often providing guidance and valuable discussions. Due to this reason, and many others, it was an extremely difficult decision to go to Edinburgh in the third year of my Ph.D and continue my research there, but soon after I would realize it would be a marvelous experience. Working with Alexandra Birch, whom I could call my honorary co-advisor, taught me so much, not only scientific but also from social terms, and opened so many doors in my academic life. I was thrilled to meet and work with everyone in the machine translation research group at the University of Edinburgh; especially Barry Haddow and Rico Sennrich, whose work on neural machine translation with subword units had been the main inspiration to my dissertation, and the researchers Kenneth Heafield, Ulrich Germann, Roman Grundkiewicz and Adam Lopez, from whom I

had received many help and feedback. Another perk of leaving Trento was meeting many people who have shown keen interest and support to my research, especially Wilker Aziz, who helped me realize the most intriguing idea I had for many years, a novel stochastic model of morphology, which would become the concluding work of this dissertation; and Orhan Fırat, who has provided me with deeper knowledge on the practical details of designing neural models. I feel blessed to have had the chance to study with such amazing mentors.

Being away from home is not easy, even though you are following your dreams, and I would not have survived it without the people in Trento who had become a second family to me; Mattia Di Gangi, who had soon after joining our lab became a great friend and a collaborator in my research, and my dear friends Alessandra Cervone and Daniele Bonadiman, who have become, as of today, a real family. As if I was not lucky enough, I would later gain another family in Edinburgh, consisting of Janie Sinclair, Maria Sinziana and Ludovica Vissat, with whom I had the chance to experience the deepest level of friendship. I am grateful for the chance of always having lovely co-workers, my officemate of three years in Trento, Rajen Chatterjee, who had helped me so much in the beginning of my Ph.D., and my officemates in Edinburgh, Clara Vania and Sameer Bansaal, the work of whom had been so inspiring to me.

Besides all, I am thankful to my father, for being my biggest supporter in becoming a scientist, and to my mother, for always inspiring me to believe in myself and pursue everything I want to do. Having dreams, on the other hand, is not sufficient in realizing them, and requires the strongest form of persistence, which is a feature I owe to my grandmother. That is why this thesis is dedicated to her memory.

Abstract

State-of-the-art neural machine translation systems typically have low accuracy in translating rare or unseen words due to the requirement of using a fixed-size word vocabulary during training. In addition to controlling the model complexity, this limitation is also related to the difficulty of learning accurate word representations under conditions of high data sparsity. This problem is an important bottleneck on performance, especially in morphologically-rich languages, where the word vocabulary tends to be huge and sparse. In this dissertation, we propose to solve the vocabulary limitation problem in neural machine translation by integrating morphology learning within the translation model, aiding to learn richer word representations in terms of phonological and morphological information. Our model improves the accuracy while translating into low-resource and morphologically-rich languages and shows better generalization capability over varieties of languages with different morphological characteristics.

Keywords

neural machine translation, unsupervised machine learning, computational morphology

Contributions and Joint Work

The research on linguistically-motivated vocabulary reduction described in Chapter 5 has been advised by Matteo Negri, Marco Turchi and Marcello Federico, while we worked with my advisor Marcello Federico in the project presented in Chapter 6 on compositional word representations for neural machine translation. The research following in Chapters 7 and 8 was carried out during my visiting period at the University of Edinburgh under the supervision of Alexandra Birch. The idea of implementing the hierarchical decoding model and comparing it to the performance of subword-level neural machine translation has become a project at the Machine Translation Marathon in 2018, based on my proposal, where Orhan Fırat contributed with practical suggestions on the design and experimental settings, leading to the work described in Chapter 7. The project was later extended to integrate a stochastic morphology model into the decoder, which can be found in Chapter 8, where the formulations were supported by Wilker Aziz. Mattia Di Gangi has provided technical support for the implementation of character-level encoding and decoding models in Pytorch, specifically the models described in Chapters 6 and 7.

Publications

Ataman, D., Negri, M., Turchi, M. and Federico, M. (2017) Linguistically-Motivated Vocabulary Reduction for Neural Machine Translation from Turkish to English. The Prague Bulletin of Mathematical Linguistics 108, European Association for Machine Translation, Prague, Czech Republic, pp. 331-342.

Ataman, D. and Federico, M. (2018) An Evaluation of Two Vocabulary Reduction Methods for Neural Machine Translation. In Proceedings of the 13th Conference of The Association for Machine Translation in the Americas. Boston, United States. pp. 97-110.

Ataman, D., Di Gangi, M. and Federico, M. (2018) Compositional Source Word Representations for Neural Machine Translation. Proceedings of the 21st Annual Conference of the European Association for Machine Translation. Alacant, Spain, pp. 31-40.

Ataman, D. and Federico, M. (2018) Compositional Representation of Morphologically-Rich Input for Neural Machine Translation. In Proceedings of the Annual Meeting of Association of Computational Linguistics. Melbourne, Australia. pp. 305-311.

Ataman, D., Firat, O., Di Gangi, M., Federico, M. and Birch, A. (2019) On the Importance of Word Boundaries in Character-level Neural Machine Translation. *To Appear in the Proceedings of the 3rd Workshop on Neural Generation and Translation.* Hong Kong.

Ataman, D., Aziz, W. and Birch, A. (2019) A Latent Morphology Model for Open-Vocabulary Neural Machine Translation. *(Under Review)*

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine translation is the task of automatizing the process of translating text across different languages. From a cognitive perspective, this task implies being able to understand a sentence in a given source language and to generate a grammatically acceptable sentence with the same meaning in a given target language. Both its scientific challenges and potential applications, have made machine translation one of the most prominent and fascinating problems studied in computational linguistics and artificial intelligence, since the beginning of the 1950s.

Early approaches to developing machine translation systems included rule-based models built by human language experts, later replaced by statistical models, which could instead learn word or phrase-level translation patterns between two languages directly from a set of translation examples. The actual advancement of the field, however, owes itself to the very recent development of neural architectures, which provided the means to design models with advanced abstraction capacity, and allowed drastic improvements in the accuracy of machine translation in many languages. The main characteristic of these architectures is that they provide a stand-alone framework to directly learn a mapping between two languages, at the same time, modeling the languages themselves. On the other hand, the levels of

linguistic abstraction constructed by these models, including the semantic, syntactic or morphological structure of language, are not well-understood aspects to date.

The conventional neural machine translation model which we consider in this dissertation is a sequence-to-sequence model, which learns to map the sequence of source words into a latent representation and, subsequently, to generate from it the corresponding sequence of target words, one by one. For a given set of training examples consisting of sentences in two languages, thus, the model learns statistical representations of words according to their sentence context, and makes use of these representations to predict the translation probability between sentences in two languages. For many aspects, this model is motivated by a comprehensive approach for achieving a linguistically-coherent modeling of translation, as the context of a word is known to serve as a mean to represent its semantic features. On the other hand, the practical impossibility of learning representations for the entire vocabulary of a language raises a constraint on the size of model vocabularies, leading the model to suffer in computing accurate word representations under conditions of data sparsity. For instance, this includes the case when the amount of training examples is insufficient to observe words in different context, or the case of morphologically-rich languages, where the same word can have many different surface realizations due to syntactic conditions, most of which are rarely if never observed in any set of collected examples. In fact, the role of phonology, and consequently morphology, is completely disregarded in the linguistic structure hypothesized in this model, and this is one of the possible reasons why it still cannot provide sufficient translation accuracy for many languages.

This dissertation investigates the ideal means for achieving an open-vocabulary neural machine translation model, based on the simple idea that if the model is given the ability to learn morphology, it should be able

to encode and decode new surface forms of words, without the necessity to learn or store their representations explicitly. For this purpose, we present a study on methods that can be used to learn the morphological structure of words in any given language and integrate this information into the distributed representations of sequence-to-sequence models, in principle, allowing the models to generalize to previously unseen input and outputs during machine translation. We start our study in Chapter 2 by presenting the details on the design and functioning of the basic neural machine translation model and its limitations in terms of learning representations of rare or unseen words, as well as the conventional solution to the problem of vocabulary limitation based on subword units. In Chapters 3 and 4, we present some introductory information on morphology and its study in the field of computational linguistics, and the details of our experimental methodology, including an evaluation benchmark consisting of languages with different morphological characteristics. Our research starts with the development of a novel linguistically-motivated subword segmentation method for neural machine translation in Chapter 5, which is essentially an unsupervised morphology learning algorithm. The comparison of our method to the prominent subword segmentation approach in neural machine translation suggests the importance of preserving morphological information while learning representations of translation units. In light of the general limitations of the approach of subword segmentation, such as generalizing to languages with different morphological characteristics, and requiring tuning many arbitrary heuristics, in Chapter 6, we study a more generic solution for open-vocabulary neural machine translation. Our approach performs translation based on word representations that are learned by jointly modeling their phonological units as well as their sentence-level context. Our experiments will show that this model allows to reach better translation accuracy for all languages in our benchmark, whereas it also

has better capacity in representing rare and unseen words. In Chapter 7, we investigate the benefit of modeling phonological and lexical context through a hierarchical generative model, which translates words one character at a time, without the necessity to store their representations with an embedding table. The hierarchical decoding model reaches a performance on-par with the subword level neural machine translation models, at the same time experiencing difficulties in modeling the lexical context in languages with a high level of data sparsity. In order to improve the performance of the model under low-resource settings, in Chapter 8, we explore the idea of integrating stochasticity into the generative model, where each word is generated based on a set of shared latent morphological features. The stochastic hierarchical decoding model demonstrates better capability in generating unseen morphological realizations, and improves the translation accuracy in many of the morphologically-rich languages.

Chapter 2

Neural Machine Translation

2.1 Introduction

This chapter presents a brief description of the conventional neural machine translation model, including different types of artificial neural networks used in the architecture, its typical interpretation in practical cases and the limitations of the existing approaches.

2.2 Artificial Neural Networks

Artificial neural networks were developed as computational models inspired by the sensory processing system of the human brain. An artificial neural network consists of computational units called **neurons**. Similar to the stimulation of neurons in the brain during the transmission of electrical information, each artificial neuron passes a received signal at its input to its output depending on the value of an activation function.

Artificial neurons are connected to each other to form a **network**, or a computational graph, which can be used to learn an unknown mapping between inputs x and outputs y, by finding within a class of functions

$$\hat{y} = f(x; \theta) \tag{2.1}$$

the parameter values $\hat{\theta}$ which result in the best approximation with respect to a set of observations $D = \{(x_i, y_i) : i = 1, .., n\}$.

There has been a vast amount of work on artificial neural networks which has led to the development of many different types of architectures deploying different formulations for the artificial neurons and activation functions. Two of these architectures, the **multi-layer perceptron** and the **recurrent neural network**, constitute the main components of the neural machine translation model studied in this dissertation, which are used to learn different information related to the translation task and jointly

predict the mapping between an input sentence and its translation. In some extensions of the neural machine translation model, **convolutional neural networks** have also been used in order to compute input representations of translation units. The next sections present descriptions of these three architectures.

2.2.1 The Multi-layer Perceptron

The perceptron [83] is the simplest artificial neural network structure which has shown the capability of approximating linear functions. It consists of a set of neurons which apply a linear transformation on the input and a threshold function to predict the corresponding output.

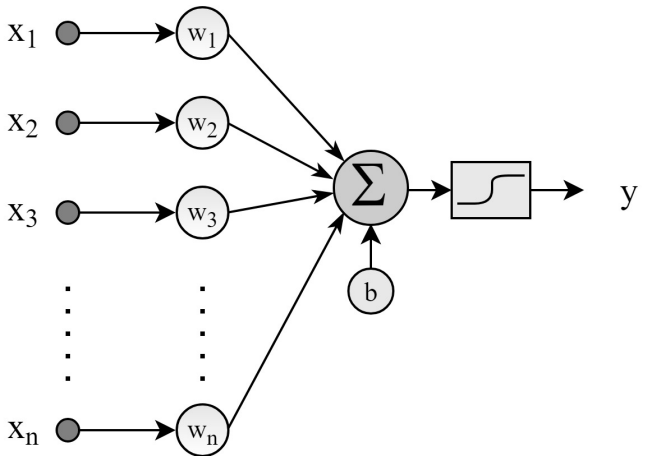


Figure 2.1: The perceptron

For a given input sequence $\{x_i : i = 1, .., n\}$ to the network, the output $\hat{y}$ is computed as a linear transformation of x followed by the activation function $g(.)$

$$\hat{y} = g(\sum_{i}^{n} w_i x_i + b) \tag{2.2}$$

where w and b are a set of scalar weights and a bias used in the linear transformation.

In order to approximate non-linear functions, it usually suffices to add one or more hidden layers between the input and output layers of the perceptron. The neurons of this layer are called hidden units, as their activations cannot be directly observed. This structure is called a **multi-layer perceptron**. Figure 2.2 illustrates the connections of artificial neurons in a multi-layer perceptron which has a single hidden layer, n inputs and a single output.

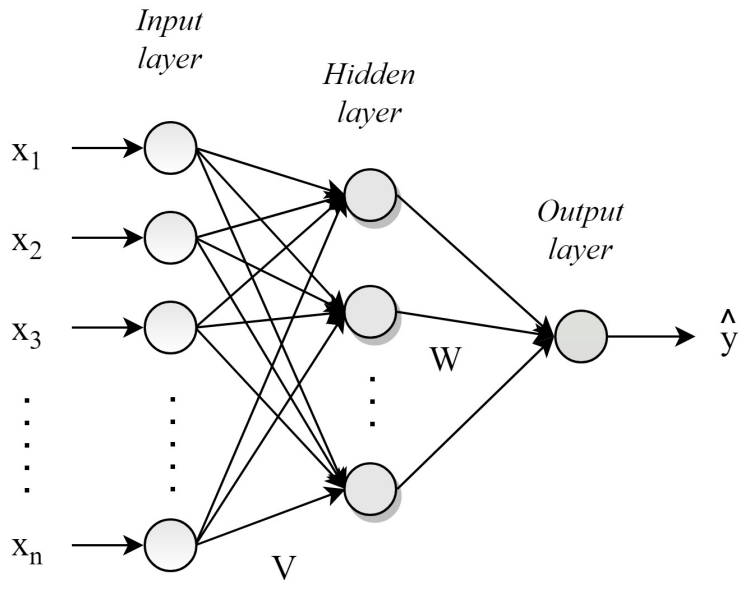


Figure 2.2: A multi-layer perceptron with a single hidden layer

In this case, the overall response of the neurons in the hidden layer h is computed through a linear transformation of the input x with the weight

matrix V, the bias vector b and a non-linear activation function $g_h(.)$[1].

$$h = g_h(Vx + b) \tag{2.3}$$

$$o = wh + c \tag{2.4}$$

$$\hat{y} = g_y(o) \tag{2.5}$$

Similarly, the response of the neurons at the output layer o is computed using the input from the layer h, the weight vector w and the scalar bias b, which is then transformed with the activation function $g_y(.)$ to obtain the prediction $\hat{y}$.

An activation function is usually a non-linear differentiable transformation that should normalize the output values into a given range. The neurons deployed in the hidden layers are typically activated using sigmoid functions such as the **hyberbolic tangent** (*i.e.* tanh(.)) function, which has values in the range (-1, 1).

$$\tanh(z) = \frac{e^{2z} - 1}{e^{2z} + 1} \tag{2.6}$$

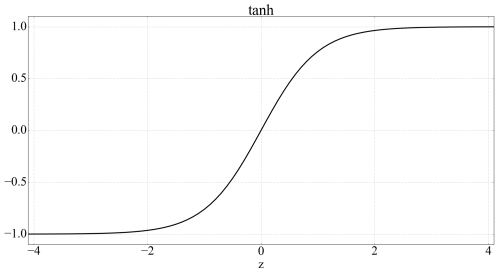


Figure 2.3: The tanh activation function

The choice of the activation function in the output layer varies depending on the specific learning task. In order to model binary output distribu-

[1]The activation function is generally a pointwise function which is applied individually on each element of the input sequence, such that $g([x_1, ..., x_n]) = [g(x_1), ..., g(x_n)]$.

tions, a typical choice is the **logistic sigmoid** function (*i.e.* $\sigma(.)$), which has values in the range (0,1).

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.7}$$

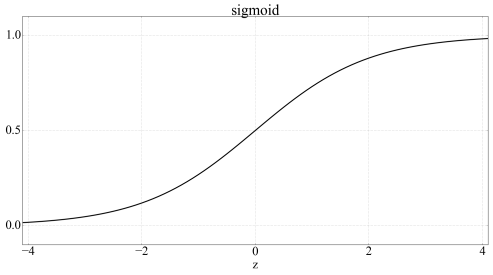


Figure 2.4: The logistic sigmoid activation function

If the output nodes are interpreted as probabilities of a set of categories, a conventional choice of activation function is the **softmax**, defined as:

$$\text{softmax}(z)_j = \frac{e^{z_j}}{\sum_k e^{z_k}} \tag{2.8}$$

The softmax function normalizes a K-dimensional vector z with arbitrary real values into a same size vector of probabilities, where K is the number of categories.

$$\text{softmax}(z)_j > 0, \forall j \tag{2.9}$$

$$\sum_k \text{softmax}(z)_k = 1 \tag{2.10}$$

On the other hand, if the output should predict a set of, for instance, positive or non-zero real numbers, one can opt for a rectifier, or specifically, a **rectified linear unit** (*i.e.* ReLU(.)), which converts all output values to positive, or a **softplus** activation, which ensures non-zero and positive

values.

$$\text{ReLu}(z) = \max(0, z) \tag{2.11}$$

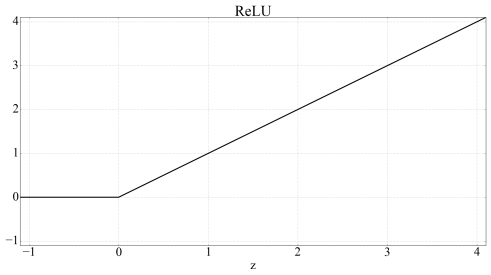


Figure 2.5: The rectified linear unit activation function

$$\text{softplus}(z) = \log(1 + e^z) \tag{2.12}$$

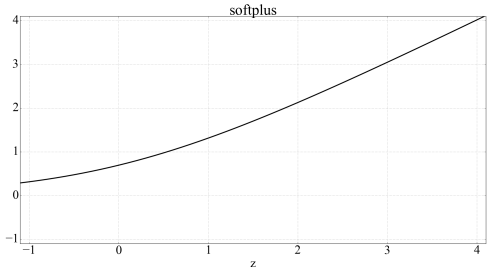


Figure 2.6: The softplus activation function

The multi-layer perceptron is a universal function approximator [30], which makes it very suitable for creating mathematical models by regression and classification analysis. Today, it is one of the most commonly used machine learning methods for supervised classification tasks. However, one major drawback of using multi-layer perceptrons is the requirement of pre-processing the input before feeding to the network, in order to

extract adequate features which might be useful for the given task. This limitation has led to the development of convolutional neural networks, which are feed-forward neural networks that integrate feature extraction into the neural network architecture and therefore eliminate the need for pre-processing.

2.2.2 Convolutional Neural Networks

Convolution is an operation that determines how two real-valued functions, x and w, overlap while w is shifted over x. It is denoted with the $*$ symbol and defined in discrete time as follows:

$$s(t) = (x * w)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)w(t-\tau) \tag{2.13}$$

In a convolutional neural network, this operation is used to predict a set of parameters w, also called as the **kernel**, from an input function x, that are beneficial in a prediction task. A convolutional neural network typically consists of an input and an output layer, as well as multiple hidden layers which include convolution layers, where the input is transformed into kernels using the convolution operation, activations, pooling layers, and fully-connected hidden layers. The **pooling** layer aids in filtering out the features relevant to the task and conventionally include simple operations such as taking the maximum of the kernel values or averaging them.

Convolutional neural networks allow very efficient computation by establishing a sparse connectivity between the input and output, as the output is only directly connected to the feature maps. By sharing the network parameters through the kernels, they also reduce the number of parameters to be stored. Due to these reasons, convolutional neural networks are widely adopted methods in image processing applications [41].

The multi-layer perceptron and the convolutional network are **feed-forward neural networks**, as the information flows from x, through the

intermediate computations used to predict f, to the output y [41]. A major limitation of using feed-forward networks is the requirement to operate on finite-dimensional input and outputs. In order to process sequential input with varying lengths, such as sentences, a more preferred approach has become using recurrent neural networks.

2.2.3 Recurrent Neural Networks

Recurrent neural networks are artificial neural networks which can have feedback connections within their hidden layers [84]. The outline of a recurrent neural network with input, hidden and output layers and the time-unfolded representation of the network can be seen in Figure 2.7.

Since the input is sequential, the output of a recurrent neural network is computed recursively for each time step t using the current input x_t, the network weights and the updated hidden state h_t until the end of the sequence

$$h_t = \tanh(Ux_t + Wh_{t-1} + b) \tag{2.14}$$

$$o_t = Vh_t + c \tag{2.15}$$

$$\hat{y}_t = \text{softmax}(o_t) \tag{2.16}$$

where U, V and W the weight matrices at the input, recurrent and output layers, respectively.

Recurrent neural networks have achieved very successful results in many tasks involving learning structural dependencies between sequential variables, and eventually became the core of the state-of-the-art architectures in machine translation.

The next section presents details on the optimization of the parameters of artificial neural networks.

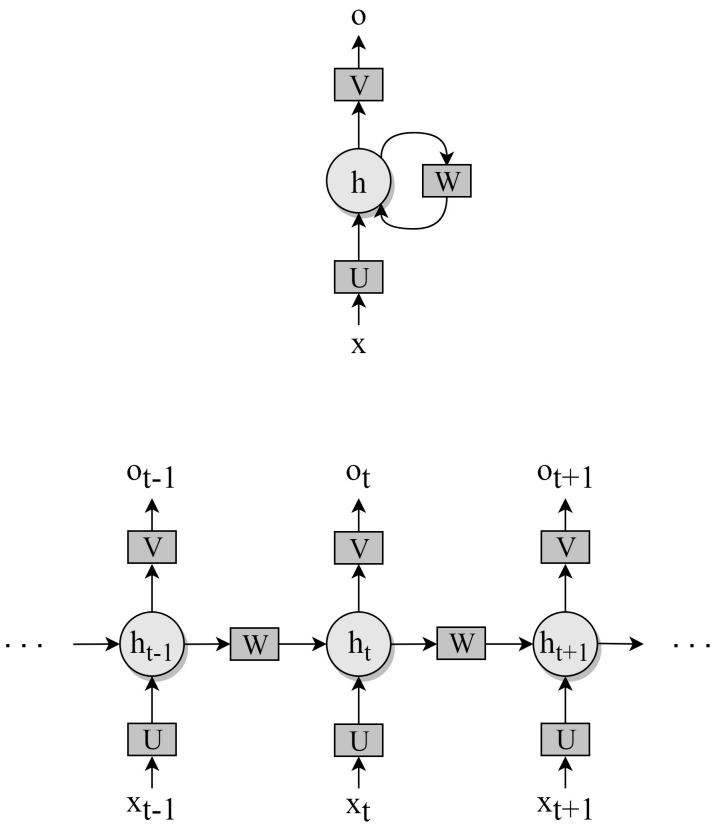


Figure 2.7: A recurrent neural network and its time-unfolded representation

2.2.4 Training and Optimization

The parameters θ of an artificial neural network, consisting of the weight and bias values at each layer, are jointly optimized to find the best function approximation $\hat{y} = f(x;\theta)$ over a training set D, consisting of a set of observations of inputs x and their corresponding correct outputs y. The optimization is performed with a cost function $J(\theta)$ which measures how good the model predictions are:

$$J(\theta) = E_{(x,y)\sim\hat{p}_D} L(f(x;\theta), y) \tag{2.17}$$

where L is a loss function typically chosen as a distance metric between each prediction value and the corresponding correct output, and $\hat{p}_D$ is the empirical distribution obtained from the training data.

The optimal values of the parameters are predicted as the values where the the cost function $J(\theta)$ is minimum, *i.e.* the gradient of the cost function is zero. Nevertheless, computing a closed-form representation of the gradient of the cost function with respect to all the parameters of the nodes in the network is mathematically intractable. Alternatively, a direct evaluation of the gradient can be computed using the **Back Propagation** method, or its extension for recurrent neural networks, **Back Propagation Through Time** algorithm [84].

Each computation of $\hat{y}$ from an input x using the artificial neural network is called a **forward** propagation. During training, the forward propagation continues until it produces a scalar cost $J(\theta)$. In order to compute an estimation of the gradient of the cost, the same network can be used to perform a **backward** propagation, where the gradient of the cost function computed at the last nodes can be propagated backwards until the input nodes to obtain an evaluation of the gradient with respect to the parameters of each layer. At each layer, the gradient of the output with respect to all node parameters are computed according to the chain rule of derivation.

The evaluation of the gradient can then be used to find the optimal values of the network parameters which minimize the cost function. This is typically accomplished using gradient-based optimization methods, the most well-known of which is the **stochastic gradient descent** [12]. Stochastic gradient descent is an iterative method for updating the values of the parameters until they reach some optimal values. At each iteration, the algorithm randomly samples a set of n examples, called a **batch**, from the training data and computes the gradient of the objective function over this set in order to update the network parameters using the following equation:

$$\theta := \theta - \eta \frac{1}{n} \sum_{i}^{n} \nabla J_i(\theta) \tag{2.18}$$

where $J_i(\theta)$ is the value of the cost function evaluated on the i^{th} example, and η is the **learning rate**, which quantifies the amount of update to be performed.

In stochastic gradient descent, all network parameters are optimized using the same learning rate, which might lead to difficulties in reaching convergence, especially when the training data is sparse. In order to improve the performance of the optimization method, a better approach is to deploy an adaptive learning rate. One of the methods that adopt this approach is **AdaGrad** [32], the Adaptive Gradient algorithm, which deploys an adaptive learning rate depending on the sparseness of the parameters. Another commonly used method is **Adam** (Adaptive Moment Estimation) [53], where the learning rate is dynamically computed based on the values of the parameters, the gradients and their second moments.

Once the network is trained using the gradient-based optimization algorithms, one can evaluate its performance on a test data consisting of new examples. In addition to minimizing the error on the training data, the second goal during training is to minimize the difference between the training and test errors, which indicates that the model is able to predict outputs of previously unseen inputs.

If the error on the training data is small, but it is large on the test data, this might indicate overfitting, which occurs when the parameters are tuned to predict the function $y = f(x, \theta)$ too closely or exactly to match the examples in the training data. A conventional method adopted in artificial neural networks in order to prevent this is **dropout** [99]. During gradient-based optimization, the parameters of individual nodes may develop complex co-adaptations in order to improve the errors computed by other nodes. The idea of the dropout method is to randomly disconnect some of them from the network during training of each single input in order to avoid correlations between single inputs and single nodes and

force distributed learning.

2.2.5 Extensions to Recurrent Neural Networks

Empirical studies on recurrent neural networks have shown that they are prone to forgetting information relevant to the hidden state representations computed at time steps far back in history [46, 9]. This problem is also called as the **vanishing gradient problem**, and it is related to the fact that while training the network on long input or output sequences, the gradients computed with back-propagation may become smaller at each iteration. If the gradients eventually become close to zero, this effectively prevents the network weights from changing their values. In order to alleviate this problem, some studies suggested modifying the recurrent neural network architecture to allow it to learn better long-term dependencies. The most widely used models today include the **long short term memory** and the **gated recurrent unit**.

Long Short Term Memory

Hochreiter and Schmidhuber [46] suggested an extension of the recurrent neural network model where the hidden states are computed using three gates, called as the **input**, the **forget** and the **output** gates.

At each time step, the forget gate determines how much information to be passed to the next hidden state, using the input and the previous hidden state representation:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.19}$$

On the other hand, the input gate determines how much information from the input and the previous hidden state will be used for updating the

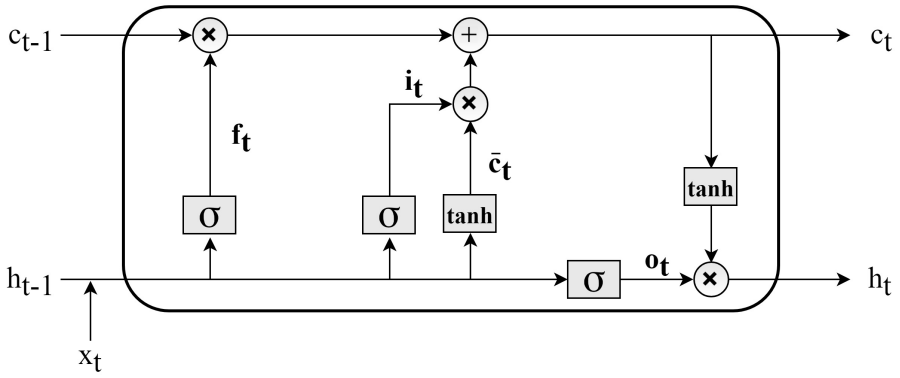


Figure 2.8: Long short term memory

network:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.20}$$

$$c_t = f_t \cdot c_{t-1} + i_t \tanh((W_c[h_{t-1}, x_t] + b_c) \tag{2.21}$$

where $\hat{c}_t$ represents the **cell** state, a novel component which accumulates information about the previous computations. The cell state is then passed to the output gate, which has the role of filtering how much information from the cell is going to be used to produce the new hidden state of the neural network:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.22}$$

$$h_t = o_t \cdot \sigma(c_t) \tag{2.23}$$

The long short term memory network has significantly better performance in learning long-term dependencies, whereas it also increases the computational cost of the recurrent neural network in terms of training time and number of parameters. A less computationally demanding alternative is the gated recurrent unit.

Gated Recurrent Unit

The gated recurrent unit [16] was proposed as an alternative solution to the vanishing gradient problem. It has a **reset** and an **update** gate which determine how much information to be eliminated (or forgotten) in the next iteration, and how much of it will be used to update the network weights.

$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \tag{2.24}$$

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \tag{2.25}$$

The next hidden state is computed using the values of the reset and update gates, the previous hidden state and the input as follows:

$$\overline{h}_t = \tanh(W[r_t h_{t-1}, x_t]) \tag{2.26}$$

$$h_t = (1 - z_t)h_{t-1} + z_t\overline{h}_t \tag{2.27}$$

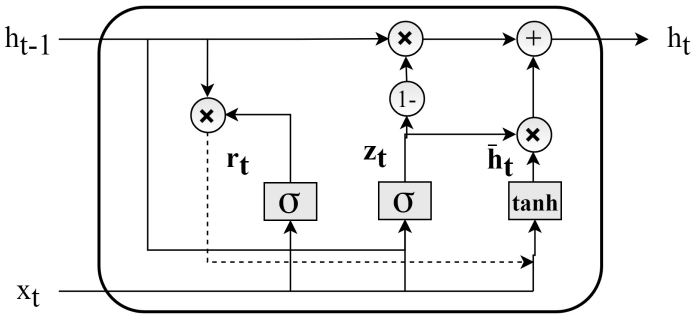


Figure 2.9: Gated recurrent unit

Due to the less number of parameters, the gated recurrent unit is more computationally efficient than the long short term memory, but also typically provides worse performance. Nevertheless, both architectures are found to achieve comparable performance when an equal number of parameters are used [19].

2.3 The Sequence-to-Sequence Model

One of the earliest approaches to neural machine translation [100] is based on the idea of using a sequence-to-sequence model consisting of two recurrent neural networks, which jointly model the conditional probability of translating a source text x, represented by the input word sequence $x = (x_1, x_2, \ldots x_m)$ of length m, into a target text y, represented as the target word sequence $y = (y_1, y_2, \ldots y_i \ldots y_n)$ of length n as follows:

$$p(y|x;\theta) = \prod_{i=1}^{n+1} p(y_i|y_{i-1}, ..., y_0, x_m, ..., x_1; \theta) \tag{2.28}$$

where θ represents the model parameters and y_0 and y_{n+1} are conventional sequence delimiter symbols.

The model is essentially a language model, which learns the probabilities of words in a continuous space, based on a varying-length of context defined as all the previous words in the sentence. The input word sequence is modeled by the **encoder**, which maps each input word into a context dependent continuous representation.

The inputs of the encoder are $|V|$-dimensional one-hot vectors, *i.e.* vectors with a single bit set to one to identify each word in the vocabulary, where the total number of words in the vocabulary is $|V|$. The vocabulary is a set of the possible words in the language, *i.e.* the lexicon, and it is usually learned from the training corpus. Using a linear transformation, each one-hot vector is mapped to a more dense representation in continuous space and a lower dimension N, called an **embedding**. Using the embeddings of each word in the sentence, the encoder obtains a distributed representation of the entire source sentence, the **context vector**, using a long-short term memory [100] as in Equation 2.23, where the context is represented by the last hidden state h_m of the encoder.

A second recurrent neural network, called **decoder**, models the target

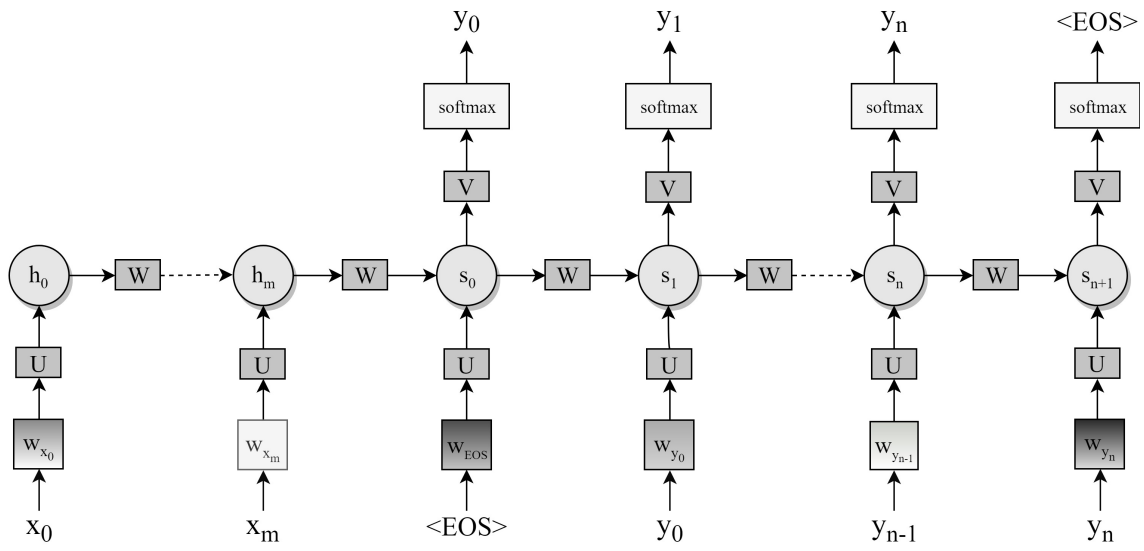


Figure 2.10: The sequence-to-sequence model

language in a similar fashion using the target word embeddings, the target sentence context, and the source context. Conditioning on the source context is accomplished by initializing the first hidden state of the decoder with the context vector.

The main objective of the model is to predict the probability distribution of the following target word given the previous word y_{t-1} and the current context (hidden state) s_t of the decoder. Each target word y_t is predicted by sampling from the word distribution the most likely target word in the vocabulary. The probability distribution $\hat{y}_t$ over the target vocabulary is computed by projecting the decoder hidden state into a vocabulary-size space through an embedding matrix $\overline{E}$ and by applying the softmax function:

$$\hat{y}_t = \text{softmax}(\overline{E}s_t) \tag{2.29}$$

where the softmax operates over the vocabulary size $|V|$.

The overall network is trained to maximize the log-likelihood of a parallel training corpus D consisting of sentences in the source language and

their translations in the target language:

$$L(D, \theta) = \sum_{(x,y) \in D} \log p(y|x; \theta) \tag{2.30}$$

via stochastic gradient-descent and the back propagation through time algorithm, as described in Section 2.2.4.

2.4 The Attention Mechanism

Although the above sequence-to-sequence model proved to be very effective, it had difficulty in translating long sentences [5]. This is due to the fact that the all of the information from the source sentence is compressed into a fixed-length context vector, which inevitably suffers of information loss when long sequences are encoded.

In order to increase the performance of the model in translating long sentences, Bahdanau *et al.* [5] proposed to compute the context vector dynamically with an **attention mechanism**. During decoding, the attention mechanism predicts a set of words in the source sentence where the most relevant information about the next word to be generated is concentrated. The model then predicts each target word based on the current context vector associated with these source positions and the previously generated target words. An illustration of the decoding process using the attention-based sequence-to-sequence learning model can be seen in Figure 2.11.

In the extended architecture, a bi-directional recurrent neural network is used to encode the source word representations, which allows to model the word context with both left and right neighbors. A bi-directional recurrent neural network consists of two recurrent neural networks that iterate over the input sequence in opposing directions: a **forward** recurrent neural

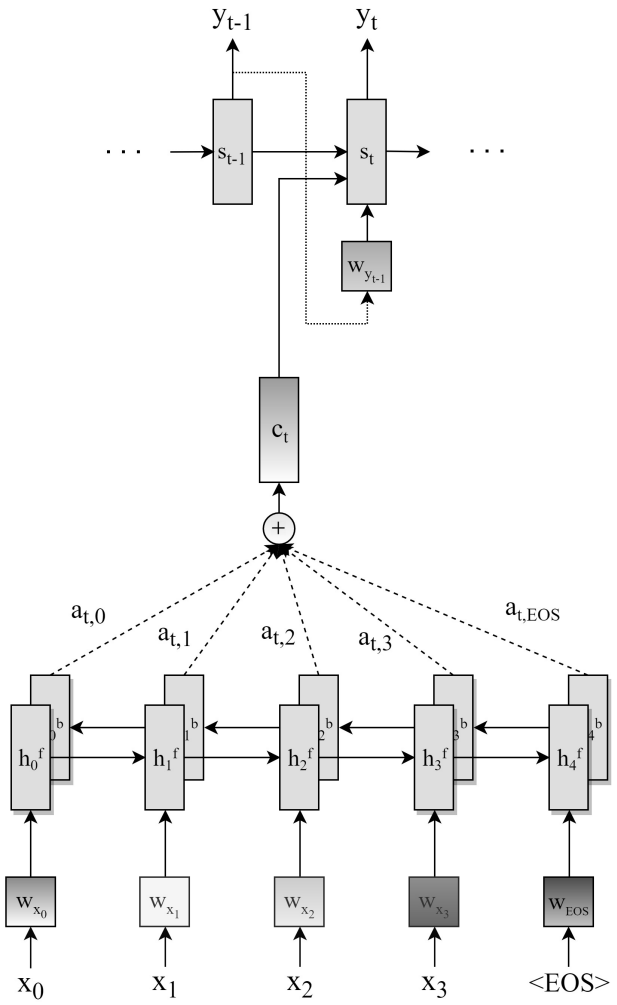


Figure 2.11: The attention mechanism

network learns the word representations given the previous context, while **backward** recurrent neural network learns the representation based on the future context, allowing to incorporate positional information of the words in the sentence. The output of the bi-directional recurrent neural network is the concatenation of the last hidden states of the two layers [91].

At each time step, the attention mechanism computes the context vector relevant to the current target word y_t as a weighted sum over the encoder

hidden states

$$c_t = \sum_{j=1}^{m} \alpha_{t,j} h_j \tag{2.31}$$

where the weights $\alpha_{t,j}$ of each annotation are computed as follows:

$$\alpha_{t,j} = \frac{e^{e_{t,j}}}{\sum_{k=1}^{m} e^{e_{t,k}}} \tag{2.32}$$

$$e_{t,j} = align(s_{t-1}, h_j) \tag{2.33}$$

The score $e_{t,j}$ computed by the **alignment model** $align(.)$ indicates how the input word at position j relates to the output word at position t and is computed based on the decoder state s_{t-1} and the j^{th} encoder hidden state h_j. The alignment model is parametrized using a multi-layer perceptron which is jointly trained with all the other components of the neural machine translation model.

While improving the translation performance by adding a notion of locality to the decoder about which part of the sentence is translated at each step, the attention mechanism slightly increases the computational cost of the overall model due to the additional multi-layer perceptron that needs to process the encoder hidden states repeatedly to generate the alignments for each target word. In order to compute the alignment scores in a more efficient way, Luong *et al.* [67] proposed a simpler attention mechanism. Instead of the multi-layer perceptron, in the so-called **general attention mechanism**, the alignment scores for each target word are computed as a linear transformation on the current decoder hidden state s_t (the decoder state is updated before the attention step) and the encoder hidden states h_j

$$e_{t,j} = s_t^T W_a h_j \tag{2.34}$$

where W_a is suitable linear transformation. The context vector c_t, computed in the same fashion as in Equations 2.31 and 2.32, is then combined

with the current decoder state s_t to produce an attentional hidden state

$$\hat{s}_t = \tanh(W_c[c_t; s_t]) \tag{2.35}$$

which is used to predict the target word using the softmax layer. In addition to providing a much more efficient solution to decoding in neural machine translation, this method has also obtained the state-of-the-art performance in the English to German machine translation task [67].

2.5 Decoding

Translation of a sentence in the source language x into a sentence in the target language y is usually referred as decoding. A naive approach to decoding is computing the translation word by word, by picking the most likely word in the vocabulary as predicted by the decoder at each time step. This approach is called **greedy search**. However, as the goal of decoding is to generate the most likely word sequence, the strategy of making word-by-word local decision is clearly sub-optimal. In other words, the most likely word at a given time may not be the best option in terms of correct meaning and grammatical accuracy of the entire target sentence.

On the other hand, generating all possible translations and performing search over an exponentially-large hypothesis space is clearly intractable. Hence, a better approach is to perform a search over a subset $\mathcal{S}(x)$ of promising candidate translations (hypotheses) and find among them the one which has the maximum posterior probability.

$$\hat{y} = \arg\max_{y \in \mathcal{S}(x)} p(y|x) \tag{2.36}$$

In this approach, the model simultaneously searches for multiple hypotheses and picks the most likely word sequence once the decoding of each hypothesis is completed. This approach is usually implemented using the **beam search** algorithm.

```
function BeamSearch(Hyp,Best,t)
NewHyp ← ()
for all (seq,score,state) in Hyp do:
      (words,logpr,state') ← ForwardPass(tail(seq),state)
      for all (w,lp) in (words,logpr) do:
            hyp=[append(seq,w),score+lp,state']
            if (IsSolution(hyp) and hyp.score > Best.score)
            then Best=hyp
            else Push(NewHyp,hyp)
      NewHyp ← Prune(NewHyp,Best)
NewHyp ← TopB(NewHyp)
if (NewHyp)
      return BeamSearch(NewHyp,Best,t+1)
else return Best
```

Algorithm 1: The beam search algorithm

The beam search algorithm performs decoding by predicting the B most likely hypotheses at each time step, which are stored in a beam of size B, by computing the joint probability of the current hypotheses and the previously generated words in each beam. The decoding continues until all beams are completed. A high-level pseudo code of the method is given in Algorithm 1.

2.6 The Vocabulary Limitation

In neural machine translation, the word embeddings which are used for the source and target words, respectively, by the encoder and the decoder are stored only for a limited number of source and target words determined by the training data. A conventional method for constructing the word vocabularies is to include only the top k frequent words in each side of the parallel corpus [5]. This limitation is due to two major aspects arising by the fundamental design of the model: high computational complexity and

inability to cope with data sparsity.

From the softmax function in Equation (2.29), one can see that the computational cost of predicting each word scales linearly with the target vocabulary size. In general, larger source and target vocabulary sizes imply an increase in the number of parameters, thus, longer training and inference time and a larger dynamic memory usage. In addition to controlling the computational complexity, using a fixed-size vocabulary for only the most frequent words is also beneficial in managing the accuracy of the word embeddings. As the translation is conventionally modeled at the lexical level, the model relies on the distributed representations of words, which can only be learned by observing the words in many varieties of context during training. Therefore, the statistical distribution of the words has a crucial role in guiding the neural machine translation model. A high level of variance in the lexical distribution implies a high level of sparsity and a low expectation to observe each individual word, resulting in low quality representations for many words in the language.

In fact, this limitation is an important bottleneck in translating any rare or unseen words that might be encountered in new sentences. One of the early approaches to improving the accuracy of the translations of unknown words was proposed by Luong *et al.* [67], which replaces the out-of-vocabulary words in the output with their corresponding inputs using a word alignment model as a post-processing step to decoding. This approach is useful for translating rare words like numbers or named entities that are not found in the vocabulary, and in principle, should have the same form in two languages. However, it is far from providing a complete solution to the vocabulary limitation problem, since a large portion of the vocabulary in a morphologically-rich language contains infrequent words that are derived or inflected word forms and carry important semantic and syntactic information, thus, cannot be translated in isolation from the rest

of the sentence.

2.7 Subword Segmentation

A better approach that addresses the vocabulary limitation in neural machine translation, which has now also become the prominent method, is redefining the model vocabulary in terms of interior orthographic units compounding the words, called as **subword units**, and learning translation as a mapping between the subword units of two languages. In order to predict a subword vocabulary, a segmentation algorithm is applied on the training corpus prior to training the neural machine translation model, which splits each word in the corpus into a sequence of subword units while preserving the original word boundaries and the splitting positions. After translation, using the splitting marks, the words in the target sentence are reconstructed from the generated subword units.

The algorithms used to apply subword segmentation generally differ by two main approaches; including likelihood-based statistical algorithms that predict a set of subwords that can optimally fit a given vocabulary size based on their frequencies and disregard any linguistic notion, and supervised morphological analysis tools which can be used to transform the surface forms of words into sequences of morphological segments or representations of their lemmas and syntactic features. The next section presents an overview of both approaches as well as their potential limitations.

2.7.1 Byte-Pair Encoding

Byte-pair encoding is a likelihood-based subword segmentation method that has obtained successful results in neural machine translation of many languages [11]. It is originally a data compression algorithm that mini-

mizes the length of sequences of bytes by finding the most frequent consecutive byte pairs and encoding them using unused byte values [34]. It was modified by Sennrich *et al.* [94] to apply vocabulary reduction for neural machine translation, where the most frequent character sequences are iteratively merged to find the optimal description of the corpus vocabulary.

This purely statistical method is based on the assumption that many types of words can be translated when segmented into smaller units, such as numbers, named entities and loanwords. This assumption holds true when applied on languages which do not exhibit high lexical sparsity, where the rare words in the training corpus are typically of this nature. Nevertheless, in case of morphologically-rich languages where common morphological paradigms such as derivational or inflectional transformations are frequently observed, the method lacks a linguistic notion that could allow it to better generalize syntactic patterns among the data and use the vocabulary space more efficiently. Table 2.1 lists some of the entries found in the neural machine translation model dictionary after the segmentation of the corpus with byte-pair encoding, which is seen to store many repetitions of the same lemma in different surface forms.

Another crucial problem is related to the semantic losses which occur due to segmenting words at positions which breaks the morphological structure. Table 2.2 presents some of the typical mistakes observed in the model output using subword units segmented with byte-pair encoding. In the first example, the Turkish word *'kanunda'* (*'in the law'*), the lemma of which is *'kanun'* (*'law'*), is segmented in the middle of the root, which causes a semantic shift. The segmented word now becomes a completely different word, *'kan'* (*'blood'*). In the second example, inaccurate segmentation of the root and suffixes leads to generate the wrong inflected form in English.

The lack of consideration on the morphological structure of words during vocabulary reduction is not only delimiting in terms of the translation

Corpus Frequency	Vocabulary Entry	English Translation
2184	hapis	*jail*
1011	hapishane	*jailhouse*
793	hapishan@@	-
587	hapishanede	*in the jailhouse*
471	hapisten	*from the jail*
245	hapishaneden	*from the jailhouse*
229	hapishanesinde	*at the jailhouse of (him/her/it)*
181	hapishanenin	*of the jailhouse*
170	hapis@@	-
156	hapiste@@	*(in the jail)@@*
149	hapisteki	*the one in the jail*
121	hapisan@@	-
100	hapishanesine	*to the jailhouse of (him/her/it)*

Table 2.1: Turkish vocabulary entries obtained after segmentation with byte-pair encoding

Source	Segmentation	Output	Reference
kanunda	kan@@ unda	in **your blood**	in **the law**
sigortalılar	sigor@@ talı@@ lar	the insure**r**s	the insure**d ones**

Table 2.2: Examples of output from the neural machine translation model trained with a Turkish byte-pair encoding subword vocabulary

accuracy, but also potentially harmful for the generalization capability of the neural machine translation model, as it might prevent the model from generating unseen word forms corresponding to combinations of different morphological paradigms applied on a given root.

2.7.2 Supervised Morphological Analyzers

The second family of approaches to neural machine translation using subword units include supervised morphological analysis tools that segment words into their morphemes or morphological features. For instance, Sánchez-Cartagena and Toral [87] suggested using the Finnish morphological seg-

mentation tool Omorfi [79] to separate words in the training corpus into their bases and inflectional suffixes to perform vocabulary reduction in English-to-Finnish neural machine translation. Similarly, Huck *et al.* [49] and Tamchyna *et al.* [101] applied morphological analysis to split words into sequences of lemma and syntactic feature sets in English-to-German and English-to-Czech neural machine translation.

Since supervised morphological analysis preserves semantic and syntactic information during segmentation, it can potentially eliminate morphological errors that may be caused when statistical subword segmentation is used. However, one major disadvantage of this approach is related to the fact that morphological analysis of a given corpus is deterministic, hence, the subword vocabulary after segmentation may be still too large in terms of the computational cost of training the neural machine translation model. In this case, the model vocabulary is usually constructed by applying cut-off thresholds on the vocabulary and reducing the coverage of the long tail of less frequent subwords, or further applying byte-pair encoding to reduce the vocabulary to a given size. In this case, one can obtain a more compact representation of subword units the vocabulary, although subword segmentation being applied after morphological analysis may cause many roots to be split, leading to morphological errors. Moreover, supervised morphological analyzers are language-specific annotated tools which are not available in many languages, hence, they do not provide an ideal approach for machine translation.

2.8 Character-level Neural Machine Translation

An alternative direction to open-vocabulary neural machine translation includes a set of approaches which perform translation directly at the level of characters without applying any form of explicit segmentation. Costa-

Jussa and Fonollosa [23] proposed processing source sentences character by character by augmenting the encoder of the neural machine translation model with a convolutional neural network, which extracts phonetic features from the input sentences and feeds these feature representations into the sequence-to-sequence model for predicting translations. Lee *et al.* [62] further extended this approach to achieve fully character-level neural machine translation by changing the decoder with a character-based one [18]. Another approach that also implements fully character-level neural machine translation based on convolutional neural networks is ByteNet [51], which performs translation in linear time steps with respect to the source sentence length.

The main problem with these approaches is that they generally disregard lexical boundaries while learning distributed representations of the input units. Indeed, these methods are more optimal for Asian languages like Chinese and Japanese which do not have explicit word boundaries [62], nevertheless, for most languages where grammar operates at the lexical level, it is controversial whether syntax and semantics, and therefore morphology, can be modeled without maintaining a context also defined at the lexical level. Moreover, using solely convolution may not be sufficient in capturing information about the relative positions of the interior units inside words, which could provide important cues about their morphological roles. More information on morphology and how it interacts with different levels of language is going to be given in Chapter 3.

Cherry *et al.* extended the approach of neural machine translation based on subword units to deploy the neural machine translation model directly at the level of characters. This approach proposes performing translation with the standard sequence-to-sequence architecture based on a vocabulary of character units, while lexical boundaries are preserved by processing the spaces between words as additional tokens in the input sentence [15].

Following this approach, the character-level neural machine translation model could reach comparable performance to subword based neural machine translation, although this would require much larger networks which require a large amount of training data and long convergence time [15]. The major reason to this requirement lies behind the fact that treating characters as individual tokens at the same level and processing input sequences in linear time increases the difficulty of learning abstract linguistic functions including the semantic, syntactic and morphological structure of the language. The increased sequence lengths caused by processing the sentences as sequences of characters also augments the computational cost despite the reduced complexity in the softmax layer.

In many languages, words are the core atomic units of semantic and syntactic structure, and their explicit modeling can be useful for properly learning the target language. Another direction of approaches to open-vocabulary neural machine translation suggested character-based translation which regards word boundaries in the translation model, where translations are generated through a hierarchical decoding procedure based on word and character level representations. Nevertheless, these approaches were generally deployed through hybrid solutions due to the requirement of observing words in varying lexical context and surface forms in order to properly model the lexical distribution and learn reliable representations. Ling *et al.* [63] proposed estimating word representations through a composition function over the character units using bi-directional recurrent neural networks, where a set of word embeddings learned from a large monolingual corpus using a language model were then set as target representations to approximate the character composition function and character embeddings. However, the isolation of the training procedures of word and character representations potentially prevents the model from learning dependencies between the phonetic and syntactic structure, which

is known to be crucial in morphology. Luong and Manning [66] proposed using a hybrid network including a word-level as well as a character-level hierarchical neural machine translation model, where unknown words in the vocabulary of the word-level neural machine translation model could be generated using the character-level neural machine translation model. In addition to still requiring to store representations of a large vocabulary of words, this approach also needs to train two different components, which increases the computational cost and the length of the decoding pipeline.

2.9 Conclusion

Neural machine translation is a useful technology which can provide competitive performance in machine translation, yet state-of-the-art approaches still suffer from inefficient handling of data sparsity mainly due to the vocabulary limitation, and consequently low accuracy in translating rare or unseen words. In order to overcome the vocabulary limitation in neural machine translation, this dissertation proposes a set of methods for unsupervised learning of morphology that can provide the model with the ability to represent rare or unseen words more accurately and increase their translation quality. The next chapter introduces the concept of morphology and presents information on how it has been studied in different fields of linguistics.

Chapter 3

Morphology

3.1 Introduction

Morphology is the study of the linguistic structure that is responsible for constructing words using the atomic semantic and syntactic units in a language. The morphological phenomena observed in a language is an important factor that has a direct impact on its lexical sparsity. In order to illustrate this aspect, this chapter introduces the basic concepts of morphology and how it has been studied in different fields of linguistics.

3.2 Morphemes and Their Types

A **morpheme** is the smallest atomic unit inside a word that carries meaning. In linguistics, morphemes have been mainly studied using two approaches, either on their functional roles in terms of the semantic and syntactic attributions to the word, or how they physically contribute to the word form.

3.2.1 Formal Morphology

Formal morphology studies the physical structure of words, specifically, how morphemes can be classified according to the position they are located in the word, or the ways in which they can be combined to form the word.

The basic unit of a language which represents lexical meaning is called a **lexeme**. The lexeme symbolizes the set of forms that a word can take through combinations with different morphemes, and it is typically observed in the **root** morpheme. The root has the most crucial role of defining the meaning and contains one of several categories (*i.e.* noun, verb, adjective, or preposition).

The physical form in which the root morpheme is observed is the **base**. Other morphemes observed in the word are generally called as **affixes**,

which are typically attached to the base to form new words. An affix that is attached to the front of the base is called a **prefix**, and an affix that is attached to the end of the base is called a **suffix**. English is a language where both prefixes and suffixes can be observed, whereas in Turkish words expand only through the attachment of suffixes.

Root	Affixed Form
accurate	in<u></u>accurate
engage	engage<u>ment</u>

Table 3.1: Examples of affixation in English

Root		Suffixed Form	
geç	*'(to) pass'*	geç<u>miş</u>	*'the past'*
geçmiş	*'the past'*	geçmiş<u>te</u>	*'in the past'*

Table 3.2: Examples of suffixation in Turkish

In very few languages like Arabic, it is also possible to observe **infixes**, types of affixes that are attached to the root within a base [75]. These changes are not accepted in the same nature as **internal changes**, which cause substitution of one non-morphemic segment for another due to case changes (*e.g.* as in the transformation of *'sing'* into *'sang'*).

Root		Suffixed Form	
ktb	*'(to) write'*	**k**u**t**i**b**	*'have been written'*
ktb	*'(to) write'*	a**kt**u**b**	*'(to) be writing'*

Table 3.3: Examples of infixation in Arabic

The form of a morpheme can vary phonetically depending on its morphological environment. This phenomenon is defined as **alternation** in

linguistics, an occasion when a morpheme exhibits variation in its phonological realization. In many languages it is typical to see morphemes undergoing phonemic changes during the composition of a new word in order to achieve phonetic harmony, although their functions or meanings do not change. One such example is **allomorphy**, the case when a morpheme can be have multiple surface forms depending, for instance, on the vowels previously observed in the root or the preceding affixes.

Root		Suffixed Form	
ev	*'house'*	evde	*'at the house'*
okul	*'school'*	okulda	*'at the school'*

Table 3.4: Examples of allomorphemes in Turkish

Another non-functional change in word structure is **consonant mutation**, where a consonant in a word is adapted to another of its phonological neighbors following the attachment of a new affix.

Root	Affixed Form
seek	sought
act	action

Table 3.5: Examples of consonant mutation in English

3.2.2 Functional Morphology

From a functional perspective, morphemes can be combined with the root to produce words mainly in two ways. **Derivational** morphemes can either alter the meaning of the root to derive new meanings, or the grammatical category of the root. After formation, the derived words become indepen-

dent items in the lexicon.

Affix	Meaning	Category	Example
un-	'not'	ADJ → ADJ	unhappy
re-	'once more'	V → V	rethink
ex-	'former'	N → N	ex-husband
-able	'ability'	V → A	understandable
-al	'relating to'	V → N	refusal
-ful	'full of'	N → ADJ	hopeful
-ize	'become'	N → V	crystalize
-ly	'manner'	ADJ → ADV	accurately
-ity	'quality'	ADJ → N	priority
-en	'creation of a state'	ADJ → V	shorten

Table 3.6: Examples of derivational affixes in English

A word can take multiple derivations, and the derivation process always occurs following the same rules. Nevertheless, a derivational morpheme cannot be combined with all roots in a given category. This restriction may depend on the linguistic origin of the root, or even its phonological properties. For example, the suffix *'en'* can be attached to the root *'mad'* to form *'madden'* but not to the root *'angry'* to form *'angryen'*.

Inflectional morphemes, on the other hand, carry information relevant to syntax. They typically ensure that the the word is transformed into a correct surface form so that the sentence is grammatically acceptable. Inflection may serve in identifying the place occupied by a word in a syntactic structure, such as a phrase or sentence, or its agreement with relevant words in the phrase [52]. Thus, the lexeme remains unaltered in any inflected form [13]. Inflection can be studied in terms of the different categories of functional morphemes that can be used for inflecting nouns or verbs.

Nouns are generally used to describe semantic objects, and they can be

inflected to specify their inherent properties. One of the obligatory categories of a noun is the **number**, which indicates if the noun is singular or plural.

Root		Inflection	
ev	*'house'*	ev<u>ler</u>	*'houses'*
ağaç	*'tree'*	ağaç<u>lar</u>	*'trees'*

Table 3.7: Examples of inflection for the number case in Turkish

Nouns can also be inflected according to **case**, which indicates their grammatical configuration in the sentence, and can be one of the following: **grammatical**, **oblique**, **nominative-accusative** and **ergative-absolutive**.

The grammatical case is used to mark the syntactic function of the noun in terms of its role as an object or a subject in a verbal phrase, whereas the oblique case marks its semantic function with verbs indicating motion.

Ev<u>i</u>	boyadık.
Object	Verb
'the house'	*'we painted'*
Ev<u>e</u>	geldik.
Adverb	Verb
'to the house'	*'we came'*

Table 3.8: Examples of grammatical and oblique cases in Turkish

The inflection of nouns to comply their cases with the verbs can be achieved either through the nominative-accusative or the ergative-absolutive systems. In Turkish, nouns are inflected through the nominative-accusative system.

Case	Inflection	
Nominative	ben	*'I'*
Accusative	beni	*'me'*
Genitive	benim	*'my'*
Dative	bana	*'to me'*
Ablative	benden	*'from me'*

Table 3.9: Examples of inflection for noun cases in Turkish

In nominative-accusative languages, the subject of a transitive verb, *i.e.* **nominative** and the object of an intransitive verb, *i.e.* **accusative**, behave similarly. On the other hand, in the ergative-absolutive system, the case used for the subject of an intransitive verb and the object of a transitive verb, namely the **absolutive** and the **ergative**, are inflected differently. In some languages, the nouns are categorized by their **gender** (or **class**), often denoted as femininity or masculinity. Words in a phrase relevant to a noun are then inflected to agree with its class.

Verbs, on the other hand, typically describe an event. The inherent inflection categories of the verb include morphemes that specify the event described by the verb. The **tense** and **aspect** functions indicate the time and progress of the action.

Affix	Function	Example
-s	present tense, 3rd person	walks
-ed	past tense	walked
-ing	progressive	walking

Table 3.10: Examples of inflection on the English verb *'walk'*

The **mood** describes whether the action is necessary, possible, permissible or desirable.

Mood	Inflection	
Condition	giderse**m**	*'if I go'*
Necessity	git**meli**yim	*'I should go'*
Possibility	gide**bil**irim	*'I may go'*

Table 3.11: Examples of inflection for verb moods in Turkish

The **conjugation** of a verb classifies it according to how the inflection will be formed along with the subset of inflectional morphemes that it can be used with.

Conjugation Class	Infinitive		2PL Present
1	nuot**are**	'(to) swim'	nuot**ate**
2	scriv**ere**	'(to) write'	scriv**ete**
3	part**ire**	'(to) leave'	part**ite**

Table 3.12: Examples of verbal conjugation for 2^{nd} Person Plural (2PL) and Present Tense in Italian

Other inflectional morphemes may determine the agreement of the verb with its object or subject, specifically in terms of **person**, **gender** and **number**; or its configurational properties, such as **reflexivization**, which is a condition when the object and the subject of the verb are the same. There are also cases where the inflectional morpheme marks the location of the word in its syntactic phrase. An example to this paradigm is the **subjunctive** mood, which is used when the verb appears in a subordinate clause which is dependent on the main clause in the sentence with a verb expressing desire or wish [52].

The number of categories that could be observed for each inflectional morpheme may vary from language to language. For instance, in German there are three noun classes, whereas in English and Turkish, there are none. Verbal classes in English are not inflected, whereas in Italian,

verbs are inflected for all persons according to the gender of the subject. Although characteristics and the possible number of derivational or inflectional morphemes greatly vary depending on the language, one can still observe a variety of generalizations about formal properties of the morphemes that are universal. For instance, if a word contains both a derivational and an inflectional affix, the derivational affix would always be located closer to the root [55], as derivation takes place before the root can be inflected. The order of the inflectional affixes are also bound to specific rules, which also depend on the language.

Another way of forming words that is seen in many languages is **compounding**, where words from different categories are combined to express new meanings. In most cases, the category of the rightmost morpheme determines the category of the entire word [75].

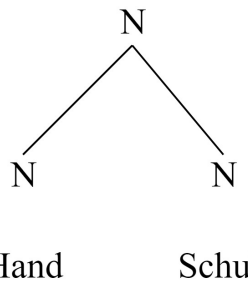


Figure 3.1: Forming of the German compound word *'Handschuch' ('glove')* from the words *'hand' ('hand')* and *'schuh' ('shoe')*

3.3 Morphological Typology

The world languages are generally grouped into two categories in terms of their morphological structure. In **analytic** languages, the morphemes are unbound, *i.e.* they can form a word on they own without the necessity to be attached to any base. In this case, there is typically a one-to-one corre-

spondence between a word and a morpheme. Many Astroasiatic languages like Thai, Vietnamese, Mandarin Chinese, Lao, Mai Brat and Khmer have analytic morphology.

เขา กำลัง เรียน ภาษา ไทย อยู่

Khaw	**kamlang**	rian	phasaa	thaai	**yuu**
S/he	PROG.	study	language	Thai	at

*She **is** study**ing** the Thai language.*

Figure 3.2: Example of verbal inflection in Thai

In **synthetic** languages, a word can contain numerous bound morphemes, which can only be used with other morphemes for inflecting or deriving new word forms. The bounding of multiple morphemes to form words can occur in three different ways. **Fusional** languages are characterized by their tendency to use a single inflectional morpheme to denote multiple grammatical, syntactic, or semantic features. When combined together, the phonetic forms of semantically distinct features are merged to create a single and unique bound form, thus, the morphemes are no longer observable individually at the surface level. The languages in the Indo-European family typically show fusional characteristics. A high degree of fusion can also be observed in the Finno-Ugric, Uralic, and Samoyedic language families.

In analytic and fusional languages, surface forms of words tend to be shorter, thus, leading to have low lexical sparsity. In **agglutinative** languages, on the other hand, combination of different morphemes occurs in a concatenating fashion, allowing a direct identification of the morpheme boundaries in the final word form. As the word forms can grow exponentially with the concatenation of new morphemes, there is generally a

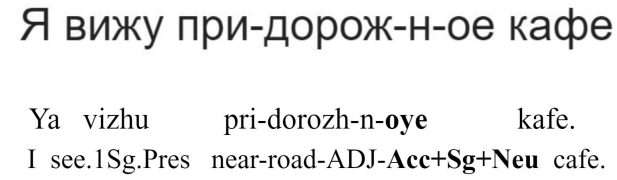


Figure 3.3: Example of noun inflection in Russian

higher level of lexical sparsity. A vast amount of word languages have agglutinative morphology, which include the Turkic, Mongolian, Korean, Japanese, and Dravidian languages in Central and East Asia, Basque in Europe, Athabascan, Siouan and Quechua languages in North America, Berber and Bantu languages in Africa.

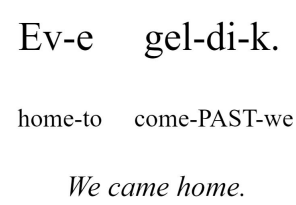


Figure 3.4: Example of noun and verb inflection in Turkish

In **templatic** languages, morphological paradigms are formed through a combination of phonological rules and functional templates. Templatic morphemes have three components: roots, patterns and vocalisms. The root is generally a sequence of three to five consonants, and vocalisms represent some of the vowels. The patterns are strings of letters which identify how roots and vocalisms are combined, and they are specific for a given inflectional or derivational transformation. A word is typically formed by inserting the root and vocalisms in the pattern morpheme. In Arabic, the majority of inflections are observed as prefixes or suffixes, whereas derivations are constructed by infixation inside the root [43]. The Semitic lan-

guages spoken in the Northern and Central Africa, such as Arabic, Hebrew, Amharic and Tigrinya exhibit templatic morphological typology. Similar to agglutinative languages, templatic languages also have high lexical sparsity.

3.4 Theories of Morphology

In linguistics, the underlying structure in language which results in morphology has been widely studied. The main schools of thought generally differ in terms of the central unit which determines the organization of the lexicon. Early studies in structural linguistics suggested that morphology should be organized in terms of a linear structure which operates on morphemes as the atomic units, where there is a direct correspondence between each morpheme and its semantic or syntactic representation. However, following work by many scholars in linguistics and cognitive sciences have shown that morphology, in reality, should have a more complex hierarchical structure in the organization of which the role of words is crucial.

3.4.1 Word-and-paradigm Morphology

One of the first theories that suggested to put words as the key units of morphological structure is the **word-and-paradigm** morphology [47, 82, 68]. First proposed by Hockett in 1954 to formalize the traditional grammar of Latin, this theory was further extended by Matthews [68] to explain inflectional phenomena observed in many languages.

In the word-and-paradigm theory, morphological transformations are organized in terms of paradigms which should be generalized between different inflected versions of words. Mainly driven by the fusional realization of inflection in many languages where inflection cannot simply be formalized in terms of rules that combine individual morphemes into words or

generate new words from roots, Matthews suggested that inflections should occur as transformations on the roots which serve as exponents of morphological feature sets.

Despite not being widely adopted in linguistics, by proving the importance of words as central units in morphological structure this approach has been quite influential in the development of more sophisticated theories of morphology [52].

3.4.2 Lexical Morphology

Similar to the word-and-paradigm theory, the lexical morphology theory also suggests that the word, rather than the morpheme, is the central organizational unit in morphology. The major claim of the supporters of this theory is that in addition to syntactic paradigms, the morphological structure of a word is also determined by the way it is pronounced. Therefore, lexical morphology is often also denoted as the theory of lexical phonology [4, 95, 78, 55, 70].

According to the theory of lexical morphology, the morphological component of grammar has a hierarchical structure with multiple layers, also called as **strata**, each of which are responsible for the application of certain inflectional and derivational paradigms. A lexical morphology model suggested by Kiparsky [55] can be seen in Figure 3.5.

The input to this model is the root of the word, to which affixes are attached as it passes from different layers. The **stratum 1** is responsible for the affixation of primary morphemes, which are described as phonologically non-neutral affixes. Non-neutral affixes are affixes which affect the location of the stress in the word pronunciation, such as the suffix *'ic'* in English.

The differentiation between neutral and non-neutral affixes has been studied in terms of strength of boundaries in English [17]. Affixation of the base with a neutral suffix is said to create a **weak boundary**, denoted

Root	Derivation
democrat	democrat<u>ic</u>
phoneme	phone<u>mic</u>
strategy	strateg<u>ic</u>

Table 3.13: Examples of derivation with the non-neutral affix *'ic'* in English

as #, whereas non-neutral affixes create **strong boundaries**, which are shown as +.

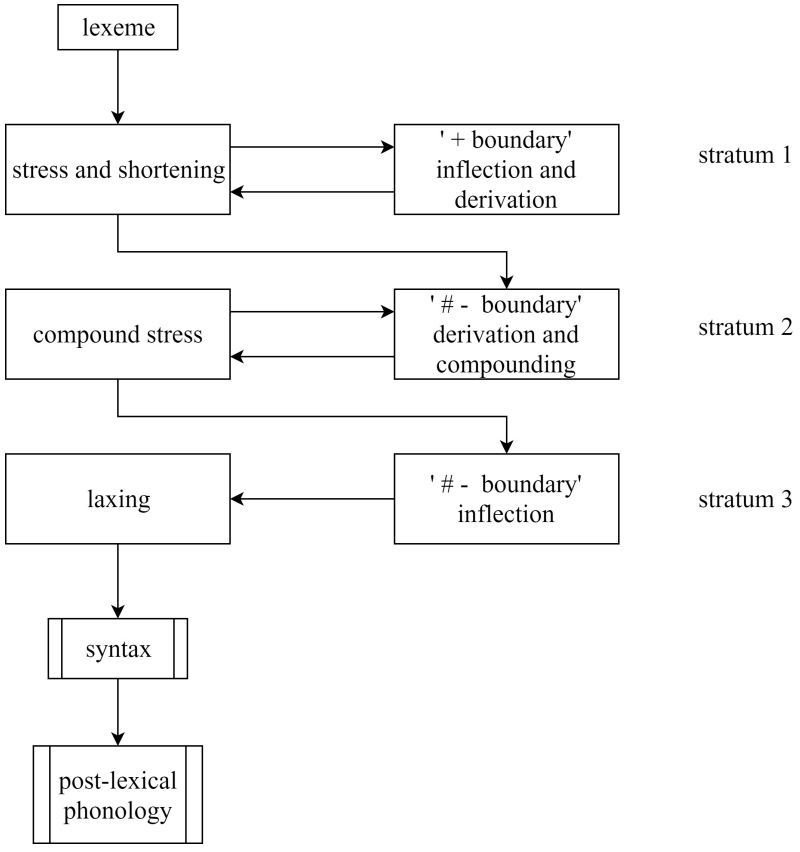


Figure 3.5: The lexical morphology model

The following layers receive as input the new derived or inflected word and applies further morphological paradigms associated with the given stratum. The affixation of neutral affixes takes place in the **stratum 2**, whereas, the inflectional affixes necessary for the grammatical role of the

word in the sentence are attached in **stratum 3**. If there are further phonological transformations of the word that are not related to word formation, these take place in the post-lexical phonology layer.

Although the hierarchical structure of the lexical morphology model has been widely accepted, there has not been an agreement on the exact number of strata that can describe all morphological paradigms. This leads to an indeterminacy on the nature of morphological rules formulated by the model and how they are organized. In lexical morphology, the morphological rules are divided into strata and can access information only related to the stratum on which they operate. However, the pre-determined ordering and the limitation of information access between strata can lead to inability to create some word forms [52]. This problem is related to the fact that of the division of strata is dependent on the categories of affixes, and not the lexeme itself. The negligence of the phonetic and categorical properties of the lexeme in determining the hiearchical structure and the final phonetic realization of the word is the major reason of the critisms [37, 98] raised to the lexical morphology model. Nevertheless, the theory still has a significant place in the study of morphology.

3.4.3 Prosodic Morphology

A **prosody** describes a phonological element that is a feature of the pronunciation of a word relevant to its morphological construction, whereas not being an inherent to the individual phonemes that form the word [52]. In contrast to the theory of lexical morphology, prosodic morphology suggests organizing a hierarchical structure of morphology in terms of prosodies.

The theory of prosodic morphology suggests a phonological model of morphology based on the generative phonology model known as **autosegmental phonology** [17, 40]. In autosegmental phonology, phonological

forms are constructed from combinations of many segments, such as stress, tone, vowel and consonants, through independent and parallel levels of representations called as **tier**s. Each tier carrying information on certain segments can interact with one another through a hierarchical structure, which is defined as the **skeletal tier** (or CV-tier) [40], although a one-to-one mapping of each element on a tier with the elements of another is not necessary; the theory suggests, on the contrary, that only relevant elements of each tier should be connected. The constraints that determine the mapping of tiers are governed by the mapping principles based on the Universal Grammar [21, 40].

The hierarchical structure through which the tiers are associated to each other is, in fact, related to morphology [52]. This has led to the development of prosodic morphology [69], which, unlike previous studies that only attempted to formalize paradigms like affixation and compounding, can formalize a more complex morphological paradigm: infixation.

As described in Section 3.3, in Semitic languages, words are formed by modification of the root internally through patterns that indicate inflectional or derivational features. In Arabic, the behaviour of vowels being inserted into consonantal roots to realize morphological phenomena can be described with prosodies. In the prosodic morphology model of McCarthy, inflection of a verb can be modeled using three **tiers**. The root tier carries the lexeme of the verb and the consonants associated with it. The skeletal tier functions as the template which defines the physical shape of the word depending on its grammatical and semantic role, whereas the vocalic melody tier determines the inflectional features, such as the tense or the mood of the verb [69]. The inflection in this model takes place as an interaction between the tiers, as given in Figure 3.6. In this model, inflection of the verb *'ktb'* in the past tense and 3^{rd} person masculine (*'he wrote'*), is achieved through the vowel *'a'*, as *'kataba'*, whereas the double usage

of the consonant *'t'* in the second inflection *'kattaba'* allows to extend the meaning to *'he cause to write'*.

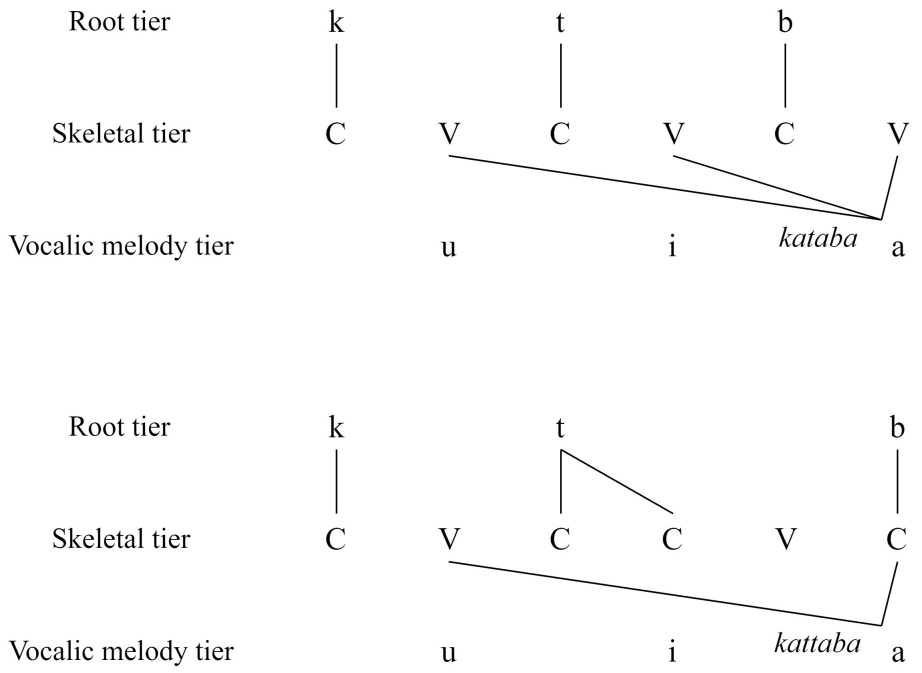


Figure 3.6: Inflection of the verb *'ktb' ('(to) write')* in Arabic with the prosodic morphology model

Prosodic morphology assumes a hierarchical lexicon which is formed through conflation of separate tiers carrying different morphemes. From this perspective, the theory has significantly contributed to morphological theory in extending previous work with a more complex and comprehensive model of morphology.

Although the formalization of a universal theory of morphology has not yet been accomplished, it is clear that morphology is one of the most complex structures in language which interacts with all different linguistic layers, including phonology, semantics and syntax.

3.5 Computational Morphology

Computational morphology aims to automatize the processing of word structure in written or spoken signals, such as identifying the morphemes and their semantic or syntactic roles in a given word, splitting the word to its stem and affixes, or predicting the corresponding surface form of a word for a given root and target inflection. Initial efforts in computational morphology considered using resources prepared by human experts such as dictionaries or lexicons and a list of rules that define the ways of performing inflection or derivation. However, unavailability of this type of data in every language and considerations on the required time and resources to be devoted in human annotation motivated the development of models that would be more independent, such as methods deploying supervised machine learning methods which can generalize morphological rules to analyze unseen words, or even unsupervised machine learning methods that can be used to directly learn morphology from natural language corpora. In spite of increasing the computationally complexity, these methods decrease the dependency to human experts and find applications in natural language processing tasks, such as machine translation [10, 22], speech recognition [2], sentiment analysis [1] and parsing [92]. This section presents the literature on computational methods for learning morphology with a general focus on the types of adopted learning approaches.

3.5.1 Supervised Morphology Learning

Supervised morphology learning is based on the hypothesis that words appear in limited variations of context and form, which can be learned and generalized into morphological rules that apply for future data. The methods which deploy supervised learning typically make use of a lexicon of root words and some examples of morphological paradigms to construct

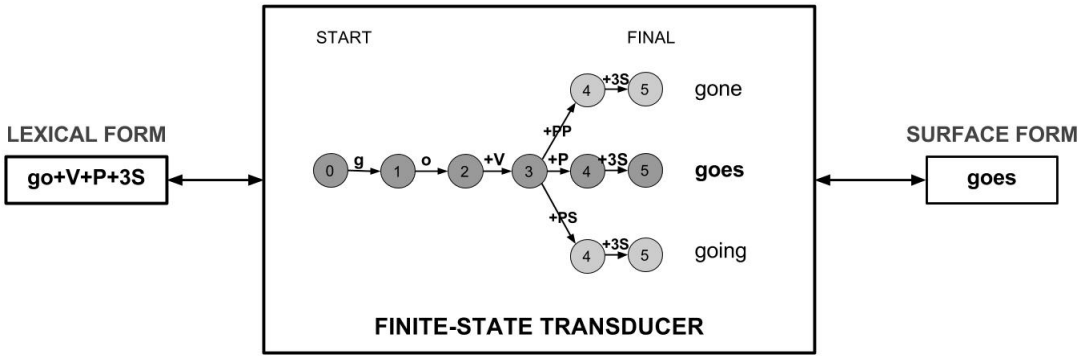


Figure 3.7: Two-level morphology with finite-state transducer

a statistical morphology model. One of the earliest studies in building non-deterministic algorithms for learning morphology was conducted by Garvin, who showed that inflectional patterns can be discovered using a list of root words and examples of their different inflected forms [35]. The sequential nature of these transformation patterns were first suggested by Golding and Thompson, who used a sequential algorithm to induce rules of inflection in English [38].

A more extensive sequential morphology induction method was developed by Theron and Cloete, which uses a hierarchical morphology model, called as the **two-level morphology** model [58] consisting of a lexicon which contains the list of morphemes in their basic forms (*i.e.* bases and affixes) and a set of inflectional transformation rules defined by finite-state automata [103]. Most morphological phenomena can be described with finite-state techniques, especially when morphological processes are assumed to occur in a concatenative fashion. The two components work together to generate and recognize new word forms in the form of two composed transducers. In this model, inflection is formulated as insertions of affixes to a given base, where the base and affixes of a word are discovered through an exhaustive search in the lexicon by choosing the most frequent segments that could be combined to obtain the word form and would result

in the minimum amount of editions. Despite its almost naive simplicity, this model can efficiently find the roots of many previously unseen words.

Finite-state morphological analyzers are high-precision tools useful for many natural language processing applications, although due to the requirement of preparing hand-crafted features and large lexicons that should contain most words in the vocabulary, they do not provide ideal solutions in terms of efficiency and usability in different languages. In order to integrate more autonomy to the rule induction procedure, a few studies suggesting using probabilistic sequential learning models, such as semi-Markov conditional random fields [25]. Semi-Markov Conditional random fields [61, 88] are discriminative un-directed graphical models based on the Markovian assumption, which model the formation of sequential structures by transitions between probabilistic states, where the probability of observing a state is dependent on the previous states. This analogy can be used for morphology learning by modeling inflectional transformations as transitions between states which represent different segments in a word. Current state-of-the-art models in morphological segmentation typically deploy more advanced learning methods, such as long short term memory networks [109] or the sequence-to-sequence learning model [33].

Compared to finite-state transducers, probabilistic sequential inference models generally have higher computational complexity, however, they allow better generalization capability. Nevertheless, for many languages, there are often not any annotated tools or labeled data available to train algorithms for morphological analysis and segmentation. The next section presents studies aimed to fully automatize morphology learning by using methods that do not require any supervision.

3.5.2 Unsupervised Morphology Learning

Methods that aim to learn the morphology of a language in an unsupervised fashion generally consider the phonological form of a word and predict its segments by relying on the frequency of each segment and some border heuristics. One of the earliest methods for unsupervised morphology learning was developed by Harris [45], which computes the frequencies of letters in a given corpus according to the varieties of their successors or predecessors, and uses these statistics to predict affixes or paradigm candidates to generate new words from characters. The most frequent character sequences are given a direct interpretation as candidates for segmentation. In his method, Harris explicitly targets the variety in letter successor types independent from their individual frequencies.

Based on the idea that the frequency distributions of characters might carry a better notion of the morpheme boundaries, following studies suggested performing segmentation based on the entropy of the character token distribution [44] or by computing the maximally-skewed characters [36] to predict the presence or absence of a morpheme boundary at a given position in the word. The calculated statistics can then be used for segmentation by comparing the frequencies of segments with pre-determined cut-off thresholds or ranges [45, 44]. As the over-representation values that determine the thresholds, many studies used the **more-frequent-than-its-length** heuristics, which assumes that a character sequence will have a higher expectancy as its length decreases.

Using threshold heuristics are simple solutions to unsupervised morphology learning; although they lack any theoretical background for generalization of morphology across the language. Moreover, they are not completely unsupervised as their settings require human expertise. These problems have led to development of more sophisticated models inspired

by compression techniques. One very successful approach in this direction include the **minimum description length** based morphological analysis, which aims to generate a description of data which minimizes its length according to a specific coding scheme [31]. The algorithm iteratively computes a set of hypotheses H, each of which is used to encode data D

$$L(D) = \underset{H \epsilon \mathrm{H}}{\operatorname{argmin}}(L(H) + L(D|H)) \tag{3.1}$$

where the hypothesis minimizing the data length is chosen as the best description of data. The main idea behind minimum description length based morphology learning is that morphology is essentially a structure that allows expressing new semantic and syntactic information by using different combinations of smaller and more compact units, each of which corresponds to a distinct semantic or syntactic feature, and morphological regularities or patterns could be discovered by searching for a set of segments which results in the most compact description of a given corpus.

The minimum description length principle has been widely adopted in unsupervised morphology learning algorithms, which generally work by first identifying the set of stems and affixes, and then running a greedy search among the lexicon and the list of word formation rules to find the shortest descriptions of the words in a corpus [89, 39, 72, 26]. Nevertheless, one problem related to these approaches is that the semantic and syntactic properties of morphemes are completely disregarded during segmentation. A better alternative is to use a stochastic sequential learning algorithm which models morphological segmentation considering properties of words other than their frequencies. The Morfessor **Categories-MAP** [27] model and its extension for semi-supervised learning, **FlatCat** [42] are algorithms which are based on this approach.

Similar to the two-level morphology model of Koskenniemi [58], the

category-based morphology model of Morfessor consists of mainly two parts: a **lexicon** that contains the list of morphemes and a **grammar** which defines the set of rules that can be used to combine different morphemes together to generate new words. The method is optimized using the Maximum A-Posteriori principle, the aim of which is to avoid overfitting by finding a balance between model accuracy and complexity [27]. The maximum posterior conditional probability of the model M given a training data D is formulated as follows:

$$\arg\max_M P(M|D) = \arg\max_M P(D|M)P(M) \tag{3.2}$$

where the two factors respectively represent the likelihood of the training data and the prior probability of the model. The former is estimated by a Hidden Markov Model which considers the transitions between different morpheme categories (*e.g.* prefix-to-stem or stem-to-suffix) when each word in the corpus is being segmented. The latter is modeled by the lexicon, which represents a list of morphemes discovered in the corpus along with their distinctive properties, and a grammar, the set of rules to combine the morphemes.

$$P(M) = P(lexicon, grammar) \tag{3.3}$$

The grammar has a deterministic structure which defines the set of categories to which the morphemes can belong: namely a **prefix**, **suffix**, **stem** or a **non-morpheme**. The probability of the grammar $P(grammar)$ is taken as equal to 1.

$$P(M) \approx P(lexicon) \tag{3.4}$$

The prior probability of a lexicon of m morphemes μ_1,...,μ_M can be formulated in the following way.

$$P(lexicon) = m!P(size(lexicon) = M) \tag{3.5}$$
$$P(properties(\mu_1), ..., properties(\mu_m)) \tag{3.6}$$

The term $P(size(lexicon) = M)$ is discarded in the original cost function to allow the lexicon to converge to a size that is optimal for the model. The remaining attributes of morphs μ_i are modeled in the following way.

$$P(lexicon) \approx m! \prod_{i}^{m} P(usage(\mu_i))\ P(form(\mu_i)) \tag{3.7}$$
$$\approx m! \prod_{i}^{m} (P(\mu_i)\ P(length) \tag{3.8}$$
$$P(perplexity_{left})\ P(perplexity_{right})) \tag{3.9}$$
$$P(form(\mu_i)) \tag{3.10}$$

The **usage** of a morpheme is related to its meaning and is modeled with its frequency, length, and the left and rightwards perplexities. The **form** of a morpheme is the set of physical properties that distinguish it from the others in the lexicon.

The **frequency** of a morpheme, denoted as $P(\mu_i)$, aids in understanding if it is a content or a functional word. The general assumption in choosing this function is based on the fact that functional words like prepositions would be observed more frequently in a given corpus compared to content words. It can be modeled using the combinatorics of choosing m possible distinct morphemes from a corpus with n tokens, which is equal to the sum of frequencies of the m word types in the lexicon.

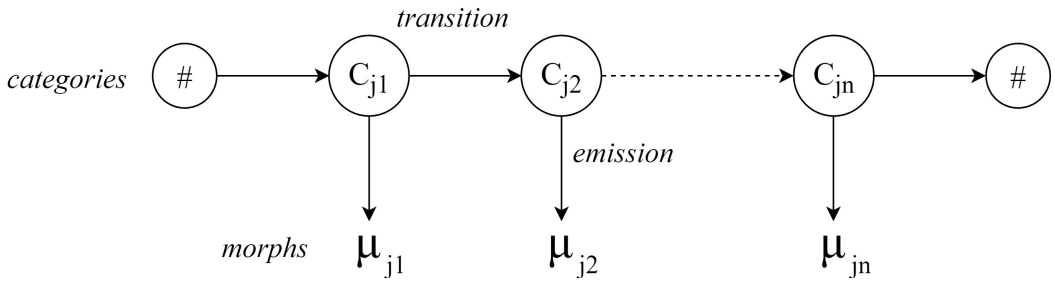


Figure 3.8: Hidden Markov Model of Morfessor Categories-MAP

$$P(\mu_i) = \frac{1}{C_{(m-1)}^{(n-1)}} = \frac{(m-1)!(n-m)!}{(n-1)!} \tag{3.11}$$

The **length** of a morpheme carries a cue on its likeliness to be a stem, whereas **perplexity** serves as a measure for predicting the immediate context and aids in estimating the probability of a morpheme being an affix. For instance, if the leftward perplexity of a morpheme is high, one might assume that the morpheme is a suffix, as it would be observed following various types of contexts. The perplexities are typically implemented as sigmoid functions.

The composition of a word is modeled with a Hidden Markov Model [8], where each hidden state represents a morphological category. The emission probability, the probability of a given morph in a category that is needed to segment the corpus, is calculated with the Bayes' formula:

$$P(\mu_i|C_i) = \frac{P(C_i|\mu_i)P(\mu_i)}{\sum_{\mu_{i'}} P(C_i|\mu_{i'})P(\mu_{i'})} \tag{3.12}$$

where

$$P(C_i|\mu_i) = P(C_i|usage(\mu_i)) \tag{3.13}$$

determines the assignment of a given morpheme to a certain category.

Given the state transition and emission probabilities, the corpus probability is computed as:

$$P(D|M) = \prod_j P(C_{j1}|C_{j0}) \prod_k P(\mu_{jk}|C_{jk}) P(C_{j(k+1)}|C_{jk}) \tag{3.14}$$

The transition probabilities between categories, as in Figure 3.8, represent the gradual formation of a word including the transition from the word beginning to the first morph until the transition from the last morph to the word boundary (#).

During training, the algorithm iteratively searches for the optimal segmentation of the corpus which minimizes the negative log-likelihood of the a-posteriori probability.

$$L(D, M) = -\log(D|M) - \log P(M) \tag{3.15}$$

Initially developed for Finnish, the Morfessor Categories-MAP model achieved state-of-the-art performance in unsupervised morphological segmentation of many languages particularly exhibiting agglutinative morphology.

3.6 Conclusion

Morphology has been a well-studied subject in computational linguistics, which led to the development of many theories and computational methods to model the structural nature of morphological transformations in language. Some of these studies, especially the lexical morphology theory and morphology learning methods, such as the finite-state transducer based morphological analyzers and the Morfessor Categories-MAP model for unsupervised morphological segmentation, have been highly influential in the development of the methods included in this dissertation. Before starting to describe the theoretical work, in the next chapter, we present the details of the experimental methodology used to confirm certain hypotheses and evaluate different approaches to open-vocabulary neural machine translation.

Chapter 4

Experimental Methodology

4.1 Introduction

In order to evaluate different approaches proposed to solve the vocabulary limitation problem in neural machine translation we construct a common benchmark consisting of languages with different morphological characteristics, and consequently, varying levels of lexical sparsity. This chapter presents the details of the evaluation methods, including the languages and the data sets used in the evaluation benchmark as well as the evaluation metrics used to measure the accuracy of the outputs of each neural machine translation model.

4.2 Languages

Most languages do not belong exclusively to one category of morphological typology. In fact, there are many languages where different morphological phenomena are observed together. Based on how much such phenomena are typical in a language, it is expected to observe increased sparseness in the lexical surface forms. Consequently, the morphological characteristics of a language would be directly influential on the statistical distribution obtained from a textual corpus in the given language.

In order to better enlighten this aspect, the evaluation of neural machine translation models in this study is performed on a common benchmark involving the translation of six languages. Each of the chosen languages in the benchmark represents a different language family and falls into a distinct combination of morphological typology. The selected languages consist of English and German from the *Germanic*, Italian from the *Italic*, Czech from the *Slavic*, Turkish from the *Altaic* and Arabic from the *Semitic* language families.

Figure 4.1: Language families included in the evaluation. Yellow: Italic, Green: Germanic, Orange: Slavic, Pink: Turkic, Gray: Semitic.

English is a West Germanic language widely spoken in the United Kingdom, Ireland, United States, Canada, Australia and New Zealand by an estimated 400 million people. It has a rather low-complexity morphology that exhibits mostly fusional and at times agglutinative transformations. We choose English as the pivot language in our machine translation experiments since the vocabulary is often small and do not necessarily imply a problem of vocabulary reduction, whereas being the most studied language in computational linguistics it stands as a reference point in comparing experimental findings.

Arabic is a Central Semitic language that is a collection of many dialects and it is spoken by around 290 million people in the world. The most widely used dialect is Modern Standard Arabic. It has a unique writ-

ing system where text is written from right to left in a cursive style. In Arabic morphology, every word is formed by a discontinuous combinations of templatic morphemes.

Czech is a Slavic language spoken by around 10 million people in the Czech Republic. It is a nominative-accusative language and, like other Slavic languages, it is rich in conjugation and nominal declension [48]. Depending on the verb, conjugation can occur as a prefix or suffix whereas the declension generally occurs through suffixes. The inflection is highly fusional, although agglutinating morphemes can also be observed.

German is a language of the Germanic family, and is spoken by around 130 million people in central Europe. German is the language with the most inflectional morphology in its family. There are three genders and four cases for nouns, and verbs are conjugated for person and number. Most inflections take place through highly fusional suffixation. It is also possible to frequently observe compounding in German.

Italian belongs to the Italic language family, which includes all Romance languages. It originated from the area corresponding to modern day Italy, and today it is spoken by about 85 million people. Italian, with its highly fusional morphology [75], presents both derivational and inflectional transformations, although its morphological complexity is quite low. Verbs are in general conjugated for person, number, gender and cases using a single suffix.

Turkish is a Turkic language and its major morphological characteristics are vowel harmony and extensive agglutination that allows a complete transparency in morphological composition. Inflections and derivations occur only by suffixation, where compounding is also rarely observed. The words always start with a root and can extend from left to right indefinitely. There are no articles and noun genders, wheres a high number of cases. In Turkish, there are about 30,000 root words and about 150

Language	Family	Morphological Complexity	Formal Types of Inflection
English	Germanic	Low	Fusional
Italian	Italic	Low	Fusional
German	Germanic	High	Fusional
Czech	Slavic	High	Mostly Fusional, Partially Agglutinative
Turkish	Turkic	High	Agglutinative
Arabic	Semitic	High	Concatenative, Templatic

Table 4.1: Languages, families and morphological typology

distinct suffixes, which can experience agglutinating concatenations and alternations, allowing the surface forms to grow exponentially [74].

4.3 Data

The training of neural machine translation models are carried out using two main approaches. The first approach constructs a common benchmark for all the chosen languages in the same domain and under low-resource settings. The benchmark includes TED Talks corpora [14], which consists of collections of speech transcriptions in generic domain. For validating and testing the accuracy of each model the official data sets of the IWSLT[1] evaluation campaign from 2010 to 2015 are used. The machine translation task is modeled by pairing each chosen language in Section 4.2 with English either as source or target, which aids in having a variety of languages with different morphological typology within the same benchmark. The details of the statistical characteristics of each training set used in the experiments and the chosen development and testing sets are given in Tables 4.2 and 4.4. In the second approach, the neural machine translation models are

[1]The International Workshop on Spoken Language Translation with shared tasks on machine translation organized between 2003-2017.

evaluated in a more challenging and practical scenario where the training data is constructed as a combination of resources from multiple domains. The data sets in the training corpora include TED Talks and additional resources from domains such as news, literature, movie subtitles and technical documents. The additional resources include EU Bookshop [96], Global Voices, Gnome, Tatoeba, Ubuntu [105], KDE4 [104], Open Subtitles [64] and SETIMES [106] corpora. Depending on the experiments, this evaluation is performed for either Italian or Turkish, which represent the least and most sparse languages in the benchmark. In order to reduce the computational cost of the experiments, the size of the resulting collected resources is reduced by data selection [29], while the vocabulary sizes on the English side are kept comparable. The statistical properties of the multi-domain training corpora are given in Table 4.3.

Language	**# sentences**	**# tokens**	**# types**
Arabic-English	238K	3,9M (Arabic)	220K (Arabic)
		- 4,9M (English)	- 120K (English)
Czech-English	118K	2M (Czech)	118K(Czech)
		- 2,3M (English)	- 50K (English)
German-English	212K	4M (German)	144K(German)
		- 4,3M(English)	- 69K(English)
Italian-English	185K	3,5M (Italian)	95K (Italian)
		- 3,8M (English)	- 63K(English)
Turkish-English	136K	2,7M (Turkish)	171K (Turkish)
		- 2M (English)	- 53K (English)

Table 4.2: Training sets from the TED Talks corpora (M: Million, K: Thousand.)

Language	# sentences	# tokens	# types
Italian-English	785K	21M (Italian) - 22M (English)	152K (Italian) - 106K (English)
Turkish-English	434K	6M (Turkish) - 8M (English)	373K (Turkish) - 135K (English)

Table 4.3: Multi-domain training set (M: Million, K: Thousand.)

In addition to machine translation, some of the proposed approaches are also evaluated in the task of morphological segmentation. In this case, different segmentation models are evaluated on the official data sets of MorphoChallenge 2010[2] [60].

All data sets are tokenized and true-cased using the Moses scripts[3] [57], except for Arabic, which is normalized using the Arabic normalization tool of QCRI[4] [86].

4.4 Evaluation Metrics

In this dissertation, different approaches for performing open-vocabulary neural machine translation are evaluated in the tasks of machine translation, as well as morphological segmentation. This allows to enlighten the capacity of each neural machine translation model to learn morphology or generalize to unseen or rare word forms. The evaluation metrics used in each task are presented in the next sections.

4.4.1 Morphological Segmentation

Morphological segmentation is essentially a classification task, where the output is drawn from a predefined set of morphological features or mor-

[2]Shared Task on Unsupervised Morphological Analysis
[3]www.statmt.org/moses
[4]alt.qcri.org/tools/arabic-normalizer

Language	Data sets		# sentences	# tokens
Arabic-English	Development	dev2010, test2010, test2011, test2012	5,835	89K (Arabic) - 114K (English)
	Testing	test2013, test2014	4,121	66K (Arabic) - 83K (English)
Czech-English	Development	dev2010, test2010, test2011	3,112	52K (Czech) - 61K (English)
	Testing	test2012, test2013	2,836	47K (Czech) - 55K (English)
German-English	Development	dev2010, test2010, test2011, test2012	5,777	108K (German) - 113K (English)
	Testing	test2013, test2014, test2015	3,543	67K (German) - 70K (English)
Italian-English	Development	dev2010, test2010,	3,517	74K (Italian) - 79K (English)
	Testing	test2011, test2012	3,230	55K (Italian) - 60K (English)
Turkish-English	Development	dev2010, test2010	2,433	34K (Turkish) - 47K (English)
	Testing	test2011, test2012	2,720	39K (Turkish) - 53K (English)

Table 4.4: Development and testing sets (M: Million, K: Thousand.)

phemes. In statistical analysis of classification, a typical measure of accuracy is the balanced F score (also called as the F_1 score).

In binary classification, the F_1 score is calculated as the harmonic mean of precision and recall as follows:

$$F_1 = \left(\frac{\frac{1}{recall} + \frac{1}{precision}}{2} \right)^{-1} \tag{4.1}$$

where precision and recall measure the percentage of the outputs to a given class that are correctly classified, and the percentage of overall correctly classified output, respectively.

$$precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{4.2}$$

$$recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{4.3}$$

4.4.2 Machine Translation

Machine translation models are typically evaluated using automatic evaluation metrics. A commonly preferred method that also shows high correlation with human judgments is **BLEU** [76]. Given a machine translated text, *i.e.* hypothesis and the correct translation *i.e.* reference, the BLEU score is a measure on the percentage of matches in words between the output and reference sentences. The n-gram precision score in BLEU is computed by summing the n-gram matches for each hypothesis in the test data with the reference sentences

$$\log p_n = \frac{\sum_{H \epsilon D} \sum_{ngram \epsilon H} Count_{matched}(ngram)}{\sum_{H \epsilon D} \sum_{ngram \epsilon H} Count(ngram)} \tag{4.4}$$

The overall precision is computed by taking the geometric mean of the n-gram precisions for each n-gram size, typically with the maximum size of 4 or 5.

$$BLEU = BP \exp(\frac{1}{N} \sum_{n=1}^{N} p_n) \tag{4.5}$$

BP is the brevity penalty which aids in penalizing the tendency to give higher precision to shorter translations. It is calculated as

$$BP = \begin{cases} 1, & \text{if } c > r \\ \exp^{1-r/c}, & \text{if } c \leqslant r \end{cases} \tag{4.6}$$

where c and r are the lengths of the hypothesis and the reference, respectively.

Solely relying on matching in the output words is not sufficient to evaluate the models, especially in the case of translating into morphologically-rich languages, where the BLEU metric could give low precision to an output which contains inflected or derived word forms which might be semantically and grammatically acceptable. Another evaluation method one can use in such a case is $\mathbf{CHR_F}$ [80], which computes the matches in the hypothesis and reference in terms of n-grams of characters, rather than words. The general formula for CHR_F score is

$$CHR_F\beta = (1 + \beta^2)\frac{CHR_P CHR_R}{\beta^2 CHR_P + CHR_R} \tag{4.7}$$

where CHR_P and CHR_R stand for character n-gram precision and recall arithmetically averaged over all n-grams and β is a weight parameter which can be used to give more favor to recall than to precision. Both precision and recall have equal importance when $\beta = 1$.

In addition to the translation accuracy, we will also test if an observed performance improvement is real or due to statistical fluctuations. We will evaluate the statistical significance of a score improvement with the Bootstrap Hypothesis Testing, as implemented in the MultEval tool [20].

4.5 Conclusion

We have presented the experimental setup which will be used to evaluate different approaches to open-vocabulary neural machine translation in this dissertation. Neural machine translation systems will be evaluated on the machine translation task using automatic evaluation metrics, whereas we also evaluate different approaches to subword segmentation on the task of morphological segmentation. All experiments are conducted in an evaluation benchmark consisting of six languages, each from a different language family and a distinct morphological typology. The next chapter starts our study by presenting a novel vocabulary reduction method based on unsupervised morphology learning.

Chapter 5

Linguistically-Motivated Vocabulary Reduction

5.1 Introduction

We start our research by investigating the primary question of whether morphological information is relevant or crucial in affecting the quality of distributed representations of translation units obtained by subword segmentation in neural machine translation. For this purpose, this chapter presents a novel statistical subword segmentation algorithm which constructs an optimal translation vocabulary while considering the morphological properties of individual subword units during splitting. Our method is an extension of the Morfessor FlatCat algorithm, where prediction of the morpheme lexicon is performed by accounting for the target lexicon size, making it useful for vocabulary reduction for neural machine translation. We first evaluate the accuracy of our method in the task of morphological segmentation against the original algorithm, and compare it with byte-pair encoding, to confirm that indeed the computed set of subword units are more coherent with morphological boundaries. We later compare the performance of our method against existing subword segmentation methods, including byte-pair encoding and the approach of vocabulary reduction with morphological analysis, in machine translation of different morphologically-rich languages using the evaluation benchmark described in Chapter 4, and perform an analysis to study the factors affecting the output quality, such as the morphological typology of the translated languages, the rate of vocabulary reduction and the size of training data used to train the models.

5.2 The Method

Linguistically-motivated vocabulary reduction is a subword segmentation method for neural machine translation based on unsupervised morphology

learning, which models the optimal way of segmenting words into subword units considering both their individual morphological properties and the required target vocabulary size. Similar to previous approaches to vocabulary reduction, the method can be used in a pre-processing step prior to training the neural machine translation model.

The model is an extension of Morfessor FlatCat [42], which uses the category-based Hidden Markov Model and maximum a-posteriori optimization as described in Section 3.5.2, and a flat lexicon structure. The category-based model is essential for a linguistically-motivated segmentation as words would only be split considering the possible categories of their subwords. In contrast to the Categories-MAP model, a distinct feature of this model is that it uses a flat lexicon structure. In a flat lexicon, the morpheme forms are encoded as strings; therefore longer morphemes that are frequent in the corpus tend to be kept unsplit due the high cost of encoding them by separate morphemes. On the contrary, the hierarchical lexicon as used in the Categories-MAP model allows to eliminate repetitions among the lexicon by using existing morphemes to describe different words. However, training of an hierarchical lexicon is not trivial especially in terms of the maximum-likelihood parameters. The frequency count of a morpheme would include its referral within the corpus as well as within the lexicon, making the two model parts, which have opposing minima, difficult to optimize together at a given rate [42]. Some analysis on the lexicon types also show that using the hierarchical lexicon, one may end up with rather conservative splitting compared to the maximum-likelihood methods [28]. Therefore, the choice of using the FlatCat model aids us in modifying the derivation of the optimization criteria.

Using the a-posteriori probability given in Equation 3.2, one can train a segmentation model considering both the model complexity and the likelihood of the corpus, without any control on the size of the output lexicon.

In order to use the model to achieve controlled vocabulary reduction for neural machine translation, we insert a constraint on the desired lexicon size into the cost function, by applying a regularization weight over the lexicon cost and giving more favor in a reduction of the model complexity during optimization. The cost function is then estimated by the general formula:

$$L(D, M) = -\log P(D|M) - \alpha \log P(M) \tag{5.1}$$

where $\alpha > 1$ would force the optimization algorithm to find a smaller lexicon size and a finer segmentation. Considering the tendency of the flat lexicon models to keep the frequent words unsegmented in the corpus [42], in order to achieve a more accurate segmentation model we disregard the frequency distribution $P(\mu_i)$ from the weighted part of the cost function. In fact, the value of the term is generally too small to affect the model complexity, but has an important role in determining the characteristics of the discovered morphemes.

We assume a linear relationship between the model likelihood and the output lexicon size, and enduce a scaling bias on the likelihood parameter to allow control over the output lexicon size by setting the regularization weight α as:

$$\alpha = \frac{m_i}{m_t} \tag{5.2}$$

where m_i is the initial vocabulary size of the corpus, measured by the number of distinct word types, and m_t is the number of morpheme types that would exist in the corpus after segmentation, *i.e.* the desired vocabulary size. The modified model has a new input parameter called as the output lexicon size, which sets the amount of regularization that should reduce the vocabulary to the desired size.

5.3 Evaluation

In order to evaluate different subword segmentation methods and verify our hypothesis that a subword segmentation which preserves morpheme boundaries of words can provide better performance in neural machine translation of morphologically-rich languages, we design a comprehensive analysis which includes an intrinsic and an extrinsic evaluation. Our intrinsic evaluation task involves morphological segmentation which compares the performance of different segmentation approaches in predicting subwords that are coherent with the morpheme boundaries inside a word. On the other hand, the extrinsic evaluation consists of evaluating each approach by means of the translation quality they obtain in neural machine translation of different language pairs and directions.

5.3.1 Morphological Segmentation

We evaluate the performance of statistical subword segmentation methods, including byte-pair encoding, linguistically-motivated vocabulary reduction as well as the original Morfessor FlatCat algorithm, in the task of morphological segmentation in Turkish by training segmentation models on the Turkish sides of the parallel TED Talks corpora and measuring the segmentation accuracy on the MorphoChallenge test sets.

As given in Table 5.1, linguistically-motivated vocabulary reduction achieves the best accuracy, also outperforming the original Morfessor FlatCat algorithm. This is probably related to the more greedy optimization preferred in the extended algorithm, which in this case results in more accurate segmentations. Lacking any linguistic notion in its statistical inference approach, byte-pair encoding obtains the lowest accuracy.

Method	**Precision (%)**	**Recall (%)**	F_1 (%)
BPE	66.17	40.51	50.26
FlatCat	77.19	40.46	53.09
LMVR	72.05	49.54	**58.71**

Table 5.1: Intrinsic Evaluation of byte-pair encoding (BPE), Morfessor FlatCat and linguistically-motivated vocabulary reduction (LMVR)

5.3.2 The Advantage of Preserving Morphological Information

We evaluate the effect of preserving morphological information during subword segmentation on the translation accuracy, by comparing the translation accuracy obtained by neural machine translation models using word or different subword-based vocabularies constructed by supervised morphological analyzers, byte-pair encoding and linguistically-motivated vocabulary reduction. This analysis is only performed for Turkish, using the morphological analyzer of Oflazer [73]. The morphological analyzer in this experiment serves as an indicator of the vocabulary reduction approach that maintains the full morphological description and semantics of the original words, nevertheless, it can only reduce the vocabulary to an extent. Hence, to eliminate the effect of the out-of-vocabulary words in the test set to the output accuracy, we set-up a controlled experiment where we segment the training set using the morphological analyzer and sample a portion of it, corresponding to approximately half of its initial size, so that the total vocabulary includes only 40,000 subword units. We sample also the development and test sets in the same fashion so there are no out-of-vocabulary words. In order to achieve a fair comparison between different vocabulary reduction approaches, we train the splitting rules of both unsupervised subword segmentation algorithms separately on the source and target sides of the parallel data.

The neural machine translation models that are used in our evaluation

are based on the Nematus toolkit [93]. They have a hidden layer and embedding dimension of 1024, a mini-batch size of 100 and a learning rate of 0.01. The model vocabulary size is 40,000 (for both source & target languages) and the subword segmentation models are trained with a number of merge rules of equal size to the model dictionary. We use the default hyper-parameters for training the Morfessor FlatCat model. The models are trained using the Adagrad optimizer with a dropout rate of 0.1 on the input layer and and 0.2 on the embedding and hidden layers. The training data is shuffled at each epoch. The models are trained until the perplexity does not decrease for five epochs and we choose the model with lowest perplexity on the development set for translating the test set. The translation accuracy of each neural machine translation model is measured using the BLEU and CHRF3 scores and significance tests are computed with MultEval, the descriptions of which can be seen in Section 4.4.2.

Table 5.2 shows the performance of different segmentation methods in the first experiment. In this setting, linguistically-motivated vocabulary reduction achieves the best performance on average, proving our hypothesis that a subword segmentation method which preserves morpheme boundaries leads to more accurate translations. Its performance is higher than byte-pair encoding by **2.2** BLEU and **1.6** CHRF3 points. Another important result of this experiment is that despite not achieving a 100 % accuracy in morphological segmentation, linguistically-motivated vocabulary reduction can still achieve comparable performance to subword segmentation based on morphological analysis. The slightly lower performance of the morphological analyzer might also be due to its lack of coverage on some of the words in the training corpus.

In order to better illustrate the properties of the subword units generated by each approach, we present example translations of two words from the test set. The two words have different roots, the first one is *'ağ'* (*'(fish-*

1. TED corpus, no-OOV case		
Method	**BLEU**	**CHRF3**
No Segmentation	17.77	0.3894
BPE	19.52	0.4233
Supervised	21.61▲	**0.4401**
LMVR	**21.71▲**	0.4390

Input (*Reference*)	**Method**	**Segmentation**	**Output**
ağlarını	BPE	ağ@@ larını	the cry
(*the nets*)	LMVR	ağ +larını	the nets
	Supervised	ağ +Noun + A3pl	networks
ağlamayacak	BPE	ağ@@ lamayacak	will not survive
(*would not be crying*)	LMVR	ağlama +yacak	will not cry
	Supervised	ağla +Neg +Fut +A3sg	will not cry

Table 5.2: Results of the 1^{st} experiment in Turkish-to-English translation, no-OOV (out-of-vocabulary) case. Top: Output accuracies, where ▲indicates statistically significant improvement over the BPE baseline ($p - value < 0.05$). Bottom: Translation examples.

ing) net'), and the second one is *'ağla'* (*'(to) cry'*). Byte-pair encoding segments both words to the same root *'ağ'*, a character sequence frequently observed in root words in Turkish. As can be seen in the output of the neural machine translation model, this leads to semantic ambiguity in the embedding of the subword unit, and consequently, inaccurate translations of both words. On the other hand, linguistically-motivated vocabulary reduction can generate relevant translations to both words, which are similar to the output of the model deploying the supervised method for vocabulary reduction.

5.3.3 The Effect of Vocabulary Reduction Rate

In a second experiment, we evaluate the methods at different rates of vocabulary reduction using data containing out-of-vocabulary words in the Turkish-to-English translation direction. In Experiment 2.a, we use the TED Talks corpus and reduce the vocabulary at a rate of 4.5 (*i.e.* from

140K to 40K words), where linguistically-motivated vocabulary reduction obtains an improvement of **2.3** BLEU points over byte-pair encoding. In the more challenging case, Experiment 2.b, we use half of the multi-domain training set and decrease the source vocabulary limit to 30,000, requiring a vocabulary reduction rate of almost two times of the case in 2.a (*i.e.* from 270K to 30K words). As given in Table 5.3, linguistically-motivated vocabulary reduction still outperforms byte-pair encoding by **1.0** BLEU point. The evident tendency of the improvements to decrease as the vocabulary reduction rate is increased suggests the importance of the vocabulary size in determining the advantage of using linguistically-motivated vocabulary reduction. At settings where the vocabulary size is too small to efficiently store a set of morphemes, the algorithm starts to oversegment morphemes in order to fit into the vocabulary, starting to create morphological errors and obtain closer performance to byte-pair encoding.

	2.a TED corpus, voc=40K		**2.b Large corpus, voc=30K**	
Method	**BLEU**	**CHRF3**	**BLEU**	**CHRF3**
BPE	20.45	42.65	24.42	0.4705
LMVR	**22.76‸**	**45.36**	**25.42‸**	**0.4771**

Table 5.3: Effects of vocabulary reduction rate. ‸indicates statistically significant improvement over the BPE baseline ($p - value < 0.05$).

5.3.4 The Effect of Morphological Typology

In the third set of experiments, we evaluate the performance of different methods on the common benchmark described in Section 4.3, in the machine translation task of the six languages both in the source and target sides of the neural machine translation model. We train the models using the same settings as in the first experiment, while we use a model dictionary of 30,000 units. The Morfessor FlatCat hyper-parameters are kept

as default except for the perplexity threshold, for which we keep the default value of 10 for six languages, while for Arabic we use the value 70 as suggested by Al-Mannai *et al.* [3].

The findings of the third experiment, presented in Tables 5.4 and 5.5, illustrate how the translation accuracy of neural machine translation models using different vocabulary reduction methods vary among different languages. A first glance at the findings proves the advantage of using subword segmentation in all languages. This is mainly due to the higher reduction of sparsity that is achieved with respect to the approach of filtering out infrequent words (**Word** method). When rare words that do not fit into the limited vocabulary are segmented into sequences of subwords, resulting with a new vocabulary of frequent subword units, as each subword is observed in more types of context than the word in which it occurs. The lower data sparsity obtained by subword segmentation versus Word is evident from Figure 5.1a, which plots the corresponding token-to-type ratios of each training corpus. The significant difference in output quality observed with two different segmentation approaches tells, however, that their impact highly depends on the nature of the splitting process and the characteristics of the language on which they are applied.

An interesting outcome of our experiments is that the improvements obtained with linguistically-motivated vocabulary reduction increases proportionally to the complexity of morphology. Arabic, Czech and Turkish all have a high level of lexical sparsity. As can be seen in Figures 5.1a and 5.1b, linguistically-motivated vocabulary reduction can reduce the lexical sparsity, *i.e.* increase the token-to-type ratio, in the corpus at higher degrees by applying a more homogeneous segmentation among the corpus, indicated in the levels of increase in the average sentence lengths. Sentence lengths generally remain close to their original lengths with byte-pair encoding. One can also see that the improvements in each language are related to

the formal characteristics of subwords. All three languages have inflections or derivations that are formed in a rather concatenating fashion, compared to German and Italian, where the affixes cannot be observed independently on the surface level. This explains the success of linguistically-motivated vocabulary reduction in learning subwords that are more consistent with the morphological boundaries. In Turkish, an agglutinative language where morpheme boundaries are transparent, it is possible to achieve a complete segmentation of the morphemes inside a word. Despite the limitation of achieving a complete morphological segmentation in Arabic and Czech, where there is less transparency in terms of the morpheme boundaries, having a notion of morphology during segmentation can still allow to obtain better translations.

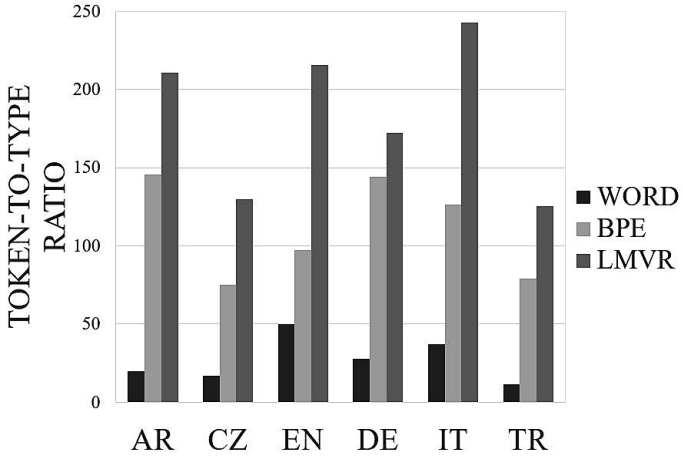
In German, English and Italian, languages with highly fusional morphology, different approaches in vocabulary reduction do not yield large differences in the output quality. This is mainly due the formal properties of fusional morphology, where typographic changes at the input may not yield sufficient information for the model to learn significantly better translations. In addition to fusional transformations, German is also rich in compounding, which can be defined as an agglutinating transformation. However, the small difference in the performance of either segmentation methods suggests that both methods can handle this phenomena similarly.

Language Direction	Segmentation (Src)	(Tgt)	BLEU	chrF3
Arabic-English	Word	BPE	26.76	0.4793
	BPE	BPE	29.59	0.5102
	LMVR	BPE	**30.67▲**	**0.5248**
Czech-English	Word	BPE	26.82	0.4689
	BPE	BPE	28.21	0.494
	LMVR	BPE	**29.2▲**	**0.5091**
German-English	Word	BPE	30.71	0.5109
	BPE	BPE	**32.57**	0.5432
	LMVR	BPE	32.53	**0.5437**
Italian-English	Word	BPE	31.41	0.5237
	BPE	BPE	**32.50**	0.5322
	LMVR	BPE	32.21	0.5302
	BPE	LMVR	32.16	0.5416
	LMVR	LMVR	**32.50**	**0.5446**
Turkish-English	Word	BPE	17.58	0.3859
	BPE	BPE	21.28	0.4335
	LMVR	BPE	22.83	0.4501
	BPE	LMVR	20.99	0.4390
	LMVR	LMVR	**23.13▲**	**0.4599**

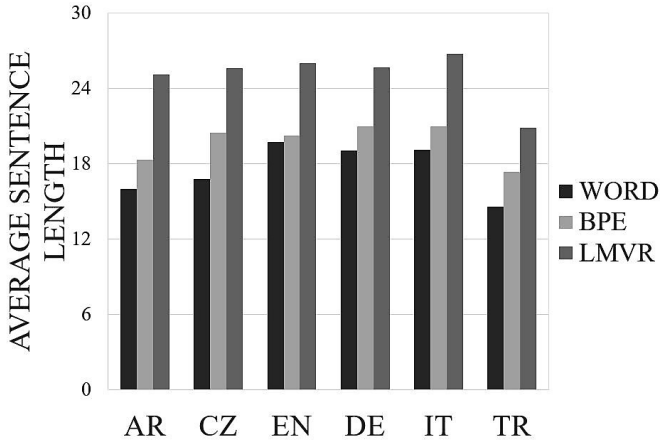
Table 5.4: Effects of morphological typology. Best scores for each translation direction are in bold font. Those marked with ▲are also statistically significantly better ($p - value < 0.05$) than the baseline.

chrF3	BLEU	Segmentation (Src)	Segmentation (Tgt)	Language Direction
0.3460	15.20	BPE	Word	English-Arabic
0.4490	17.91	BPE	BPE	
0.4610	**18.95**$^{\blacktriangle}$	BPE	LMVR	
0.3731	18.46	BPE	Word	English-Czech
0.4378	19.09	BPE	BPE	
0.4483	**19.98**$^{\blacktriangle}$	BPE	LMVR	
0.4927	26.35	BPE	Word	English-German
0.5431	27.24	BPE	BPE	
0.5485	**27.38**	BPE	LMVR	
0.5120	27.77	BPE	Word	English-Italian
0.5415	28.28	BPE	BPE	
0.5451	**28.30**	BPE	LMVR	
0.5412	27.99	LMVR	BPE	
0.5432	28.24	LMVR	LMVR	
0.2968	10.05	BPE	Word	English-Turkish
0.4183	11.31	BPE	BPE	
0.4410	12.53	BPE	LMVR	
0.4378	11.13	LMVR	BPE	
0.4435	**12.63**$^{\blacktriangle}$	LMVR	LMVR	

Table 5.5: Effects of morphological typology. Best scores for each translation direction are in bold font. Those marked with $^{\blacktriangle}$are also statistically significantly better ($p - value < 0.05$) than the baseline.



(a) Token-to-type ratios



(b) Average sentence lengths

Figure 5.1: Token-to-type ratios and average sentence lengths after subword segmentation

Another factor that affects the results is related to the statistical characteristics of the languages, which, as can be seen in the vocabulary sizes listed in Table 4.2, do not hold a large amount of sparsity. The experiment results suggest that the quantity of rare words in the vocabulary that could better be translated by different approaches could be an important indicator of the overall output accuracy. Italian, unlike German, has a morphology of comparably lower complexity and the word vocabulary is quite compact, where rare words (singletons and less frequently observed words that

are in the long tail of the frequency distribution) mostly consist of named entities or numeric expressions. This is in contrast to morphologically-rich languages, where the majority of rare words also include inflected or derived word forms. Hence, in Italian, there is less necessity for performing a morphologically-oriented subword segmentation. Hence, most of the Italian words can be kept unsegmented and translated directly at the lexical level using byte-pair encoding. Although rare inflected words may exist in the corpus, they are not observed many times, and the improvement in their translation through linguistically-motivated vocabulary reduction may not be significant enough to be observed at the output. English is also a language of this group, with a morphology of very low complexity, although most of the affixes are easily separable. The slight but significant improvements obtained with using linguistically-motivated vocabulary reduction also on the English sides of the corpus suggest that it is possible to segment the words according to their morphological boundaries, which aids in improving the translation accuracy.

5.3.5 The Effect of Data Size

We also evaluate the performance change of each method under the condition of larger size multi-domain training data in English-to-Turkish and Turkish-to-English directions. The results of this experiment, given in Table 5.6, confirm that the improvements obtained with linguistically-motivated vocabulary reduction scale to mid-resource and multi-domain settings. In the English-to-Turkish translation direction, linguistically-motivated vocabulary reduction achieves **0.85** BLEU points of improvement compared to byte-pair encoding, whereas the improvement is **0.7** BLEU points in the Turkish-to-English direction.

Language Direction	Segmentation (Src)	(Tgt)	BLEU	chrF3
Turkish-English	Word	BPE	20.13	0.4166
	BPE	BPE	24.18	0.4696
	LMVR	BPE	**24.83▲**	**0.4790**

chrF3	BLEU	Segmentation (Src)	(Tgt)	Language Direction
0.3704	11.84	BPE	Word	English-Turkish
0.4532	13.74	BPE	BPE	
0.4710	**14.59▲**	BPE	LVMR	

Table 5.6: Experiments with Turkish using multi-domain data. Best scores for each translation direction are in bold font. Those marked with ▲are also statistically significantly better ($p - value < 0.05$) than the baseline.

5.4 Conclusion

Subword segmentation is a useful approach to deal with the vocabulary limitation in neural machine translation. Depending on the morphological typology of a language, a single word may often contain multiple syntactic and semantic features, which, during translation into another language with a different morphological typology, may require learning many-to-one or one-to-many lexical mappings between multiple words. Segmenting a long inflected word into its morphemes can thus aid in distributing the information contained in the word into smaller tokens and learning a more homogeneous alignment between the translation units. Subword segmentation also allows learning higher quality of internal representations of translation units by reducing the lexical sparsity, as rare word forms are transformed into sequences of more frequent subwords, which can be observed in more varieties of context.

The experimental results of this chapter suggest that the ability of a segmentation method in capturing morphological boundaries is an important factor in determining the accuracy obtained with a neural machine translation model. Especially in the cases of high lexical sparsity occurring with agglutinative or templatic morphology, where the vocabulary generally consists of rare inflected or derived word forms, a morphological segmentation can produce subwords that not only represent a compression of a given training corpus, but also yields a statistical distribution that is coherent among collections of sentences. This is due to the fact that the domain and the size of the training data can vary in different applications, whereas with the language, its morphological characteristics stay the same. Thus, a segmentation method which can capture the morphological properties of subwords can allow for a more efficient estimation of vocabulary items which can be used to infer knowledge among different sentences and

varying corpora.

On the other hand, subword segmentation methods may also introduce many drawbacks related to their optimization, the overall computational pipeline and the fundamental design of the translation model. Statistical subword segmentation is typically deployed as a pre-processing step before training the neural machine translation model, hence, the predicted set of subword units are essentially not optimized for the translation task. Both byte-pair encoding and linguistically-motivated vocabulary reduction have hyper-parameters that require expensive tuning experiments for a given corpus and language. Our analysis shows that providing the neural machine translation model with a-priori information about morphology is useful in maximizing the translation accuracy, although a given morphology learning model is generally tailored to a specific class of languages with similar morphological typology. For instance, linguistically-motivated vocabulary reduction can successfully model agglutinative morphology, although it cannot ideally be used to learn the non-concatative morphological transformations observed in templatic or fusional morphology. Indeed, for these languages, achieving a morphological analysis solely at the surface level of words is intractable. An accurate model of morphology, however, could be established by using latent representations of the semantic and syntactic features of the morphemes observed inside words.

In regard of these considerations, the next chapter presents a novel character-level neural machine translation architecture which addresses the vocabulary limitation more efficiently by integrating morphology learning directly within the encoder of the neural machine translation model.

Chapter 6

Compositional Word Representations

6.1 Introduction

Conventional approaches which have been proposed to achieve open-vocabulary neural machine translation, including either subword or character level neural machine translation methods discussed in Sections 2.7 and 2.8, have generally modeled the translation task at the sub-lexical level, thus, the lexical boundaries are generally disregarded while learning distributed representations of the input units. Nevertheless, it is controversial whether semantics and syntax of a language can be modeled without maintaining a context defined at the lexical level. Many of the theories presented in Section 3.4 generally agree on the importance of word boundaries in defining the morphological structure in language. Moreover, the role of phonology has also been suggested to be crucial in many theoretical work.

In this chapter, we propose to implement a novel open-vocabulary neural machine translation model which hypothesizes a high-level model of morphology at its input layer, where morphological structure is estimated through a mapping between phonetic and lexical context. In this model, distributed representations of input words are learned compositionally from a vocabulary of smaller orthographic symbols, allowing the model to account for phonologically-informed word representations. Our model extends the work of Ling *et al.* [63], whereas in our approach, we learn all word representations directly from the embeddings of character n-grams, and on a set of parallel data, resulting in word representations optimized for the machine translation task. In addition to evaluating the importance of modeling translation directly at the lexical level, this approach also allows to test the possibility of eliminating the sub-optimal effects of subword segmentation. We perform a fine-grained analysis of the optimal set of vocabulary units that can be used to compose word representations in different languages, and also present an evaluation of this model against

subword and character-level neural machine translation methods, with a specific focus on their capacity in translating rare and previously unseen words.

6.2 The Method

We propose to overcome the vocabulary limitation in neural machine translation by extending the work of Ling *et al.* described in Section 2.8, where distributed representations of all words in a given source sentence are computed through morphological composition rules from a set of smaller orthographic symbols inside the words, such as character n-grams, that can easily fit in the model vocabulary. The composition is essentially a function which can establish a mapping between combinations of orthographic units and lexical meaning, that is learned using the bilingual context, so that it can produce representations that are optimized for machine translation. This approach allows to maintain the translation at the lexical level while eliminating the necessity to store word embeddings.

The mapping between the sub-lexical units and the lexical context is performed by a bi-directional recurrent neural network, which encodes the context of each interior unit inside the word and allows to capture important cues about their functional roles. As a minimal set of input symbols required to cope with contextual ambiguities, we opt to use intersecting sequences of character trigrams, as suggested by Vania and Lopez [107]. We implement the network using gated recurrent units which have shown comparable performance to long-short-term-memory units, whereas they provide much faster computation [16].

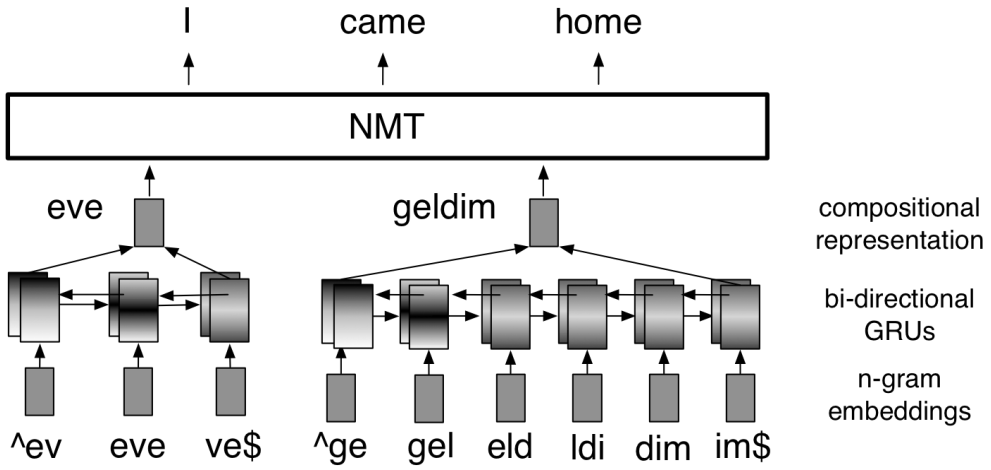


Figure 6.1: Neural machine translation with compositional input representations

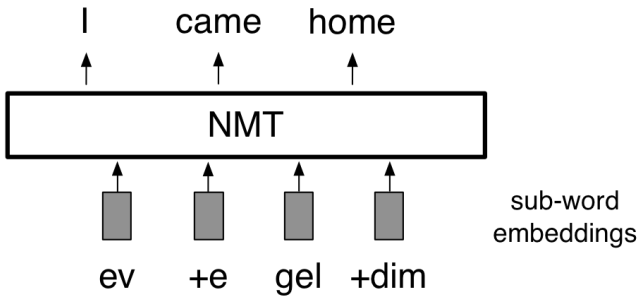


Figure 6.2: Neural machine translation with subword units

Given a bi-directional recurrent neural network with a forward (f) and backward (b) layer, the input representation $\mathbf{w_i}$ of the word x_i in the source sentence, consisting of l characters, is computed from the hidden states $\mathbf{h}^f_{i,l}$ and $\mathbf{h}^b_{i,0}$, *i.e.* the final outputs of the forward and backward recurrent neural networks, as follows:

$$\mathbf{w_i} = \mathbf{W}_f \mathbf{h}^f_{i,l} + \mathbf{W}_b \mathbf{h}^b_{i,0} + \mathbf{b} \tag{6.1}$$

where $\mathbf{W}_f$ and $\mathbf{W}_b$ are weight matrices of each recurrent neural network and $\mathbf{b}$ is a bias vector.

We implement the model by augmenting the input embedding layer of the encoder of the sequence-to-sequence learning model with a bi-directional

recurrent neural network, which computes compositional representations for all words in the source sentence. The one-hot vectors of the character n-gram units, after being fed into the embedding layer, are processed by an additional **composition layer**, which computes the final input representations passed to the encoder to generate translations. The parameters of the composition layer are jointly learned together with the the input token embedding matrix and the sequence-to-sequence model while training the neural machine translation system. Figures 6.1 and 6.2 illustrate the main difference between the approaches of neural machine translation based on compositional word representations and subword embeddings.

For an input of m tokens, our implementation increases the computational complexity of the network by $O(Kt_{\max}m)$, where K is the bi-recurrent neural network cost and $t_{\max}$ is the maximum number of symbols per word. However, since computation of each input representation is independent, a parallelized implementation could cut the overhead down to $O(Kt_{\max})$.

6.3 Evaluation

We evaluate our approach against conventional methods to open-vocabulary neural machine translation in the machine translation task of different morphologically-rich languages with varying levels of lexical sparsity. We first perform a fine-grained analysis of the translation accuracy in terms of varying levels of granularity on the input vocabulary units and morphological typology, and later we evaluate the approaches with the best performance to illustrate how the accuracy they obtain scale to high-resource settings and especially in translating unseen or rare words in the source sentences.

6.3.1 The Effect of Input Units and Morphological Typology

In order to evaluate our approach against the previously discussed open-vocabulary neural machine translation methods, we first implement it using the deep learning library Theano [102] and integrate it into the Nematus toolkit [93]. Using the TED Talks benchmark, we then perform a fine-grained analysis of the translation accuracy in terms of varying levels of granularity on the input vocabulary units. The **simple** neural machine translation model constitutes the baseline and performs translation directly at the level of sub-word units, which can be of four different types: characters, character trigrams, and subword units predicted either using byte-pair encoding or linguistically-motivated vocabulary reduction. The **compositional** model, on the other hand, performs neural machine translation with input representations composed from sub-lexical vocabulary units. In our study, we evaluate representations composed from character trigrams, and subword units segmented with statistical subword segmentation methods, including byte-pair encoding and linguistically-motivated vocabulary reduction. For each language in the benchmark, we model the translation task from the morphologically-rich language into English, where the English vocabulary consists of subword units predicted by linguistically-motivated vocabulary reduction.

In our experiments, we use a compositional layer with 256 hidden units, a one-layer bi-directional gated recurrent unit encoder and one-layer gated recurrent unit decoder of 512 hidden units, and an embedding dimension of 256 for both models. We use a highly restricted dictionary size of 30,000 for both source and target languages, and train the segmentation models (byte-pair encoding and linguistically-motivated vocabulary reduction) to generate sub-word vocabularies of the same size. We train the neural machine translation models using the Adagrad optimizer with a mini-batch

size of 50, a learning rate of 0.01, and a dropout rate of 0.1 in all layers and embeddings. In order to prevent over-fitting, we stop training if the perplexity on the validation does not decrease for 5 epochs, and use the best model to translate the test set. The model outputs are evaluated using the (case-sensitive) BLEU metric and the MultEval significance test.

The performance of neural machine translation models in translating each language using different vocabulary units and encoder input representations can be seen in Table 6.1. Compared to the results of the experiments in Chapter 5, the accuracies are generally lower due to the reduced capacity of the neural machine translation models in order to limit the computational cost of the experiments. Nevertheless, the characteristics of each approach remain consistent, which suggests that the performance of all models and the improvements would scale with increased capacity.

The experiments show that using the simple model, subword units based on linguistically-motivated vocabulary reduction achieve the best accuracy in translating all languages, with improvements over byte-pair encoding by **0.85** to **1.09** BLEU points in languages with high lexical sparsity (Arabic, Czech and Turkish) and **0.32** to **0.53** BLEU points in languages with low to medium sparsity (Italian and German). This confirms our previous results in Chapter 5. Moreover, simple models using character trigrams as vocabulary units reach much higher translation accuracy compared to models using characters, indicating their superior performance in handling contextual ambiguity. In the Italian-to-English translation direction, the performance of simple models using character trigrams and byte-pair encoding based subword units as input representations are almost comparable, showing that character trigrams can even be sufficient as the standalone vocabulary units in languages with low lexical sparsity. The findings confirm that each type of subword unit used in the simple model is specifically convenient for a given morphological typology.

Model	**Vocabulary Units**	**Input Representations**	**BLEU**				
			Tr-En	**Ar-En**	**Cs-En**	**De-En**	**It-En**
Simple	Characters	Characters	12.29	8.95	13.42	21.32	22.88
	Char Trigrams	Char Trigrams	16.13	11.91	20.87	25.01	26.68
	Subwords (BPE)	Subwords (BPE)	16.79	11.14	21.99	26.61	27.02
	Subwords (LMVR)	Subwords (LMVR)	17.82	12.23	22.84	27.18	27.34
Compositional	Char Trigrams	Subwords (BPE)	15.40	11.50	21.67	27.05	27.80
	Char Trigrams	Subwords (LMVR)	16.63	13.29	23.07	26.86	26.84
	Char Trigrams	Words	**19.53**	**14.22**	**25.16**	**29.09**	**29.82**
	Subwords (BPE)	Words	12.64	11.51	23.13	27.10	27.96
	Subwords (LMVR)	Words	18.90	13.55	24.31	28.07	28.83

Table 6.1: Experiment results in TED Talks benchmark. Best scores for each translation direction are in bold font. All improvements over the baseline (simple model with BPE) are statistically significant ($p - value < 0.05$).

Using the compositional model improves the quality of input representations for each type of vocabulary unit, nevertheless, the best performance is obtained by using character trigrams as input symbols and words as input representations. These results not only confirm our hypothesis that the translation context should be modeled at the lexical level, but also that the compositional word representations obtained by the bi-directional recurrent neural network have higher quality representations compared to those obtained from subword embeddings. Especially in the case of Turkish, where linguistically-motivated vocabulary reduction was found to provide comparable performance to morphological analyzers in Chapter 5, the compositional model seems to achieve higher capacity of learning and generalizing morphological information. Using the composition model seems to also improve the Arabic word representations, in fact, the idea of learning morphology as a function of phonetic symbols and lexical meaning is more coherent with the theory of prosodic morphology.

In order to better illustrate the characteristics of each representation, we present sample outputs from Italian and Turkish neural machine translation models in Table 6.2. In Italian, the simple model fails to understand the common subject of different verbs in the sentence due to the repetition of the same inflective suffix after segmentation. In Turkish, the genitive case *'yerlerin fotoğraflarının'* (*'the photographs of places'*) and the complex predicate *'birleştirilmesiyle meydana geldi'* (*'is composed of'*) are both incorrectly translated by the simple model. On the other hand, the compositional model is able to capture the correct sentence semantics and syntax in either case. These findings suggest that maintaining translation at the lexical level apparently aids the attention mechanism and provides more semantically and syntactically consistent translations.

Input (Simple Model)	e comunque, em@@ ig@@ *riamo* , circol@@ *iamo* e mescol@@ *iamo* così tanto che non esiste più l' isolamento necessario affinché avvenga un' evoluzione .
Output (Simple Model)	and anyway , *we* repair, and *we* mix so much that there 's no longer the isolation that *we* need to happen to make an evolution .
Input (Compositional Model)	e comunque, emigriamo, circoliamo e mescoliamo così tanto che non esiste più l' isolamento necessario affinché avvenga un' evoluzione.
Output (Compositional Model)	and anyway , *we* migrate , circle and mix so much that there 's no longer the isolation necessary to become evolutionary .
Reference	and by the way , *we* immigrate and circulate and intermix so much that you can 't any longer have the isolation that is necessary for evolution to take place .

Input (Simple Model)	ama aslında bu resim tamamen , farklı *yerlerin fotoğraf@@ larının* **birleştir@@ il@@ mesiyle meydana geldi** .
NMT Output (Simple Model)	but in fact , this picture **came up with** a completely different *place of photographs* .
Input (Compositional Model)	ama aslında bu resim tamamen , farklı *yerlerin fotoğraflarının* **birleştirilmesiyle meydana geldi** .
Output (Compositional Model)	but in fact , this picture **came from collecting** *pictures of* different *places* .
Reference	but this image **is** actually entirely **composed of** *photographs from* different *locations* .

Table 6.2: Example translations with different approaches in *Italian* (above) and *Turkish* (below)

6.3.2 The Effect of Data Size and Domain

We also investigate how the performance of each approach scale with increased levels of data sparsity through a second experiment using the multi-domain Turkish-English and Italian-English evaluation data presented in Section 4.3. We implement a more efficient version of the compositional model using the PyTorch library [77] and integrate it into the OpenNMT-py toolkit [56]. In this case, we conduct the experimental comparison between the compositional and simple neural machine translation models using an equal number of parameters, where the compositional model uses an encoder of two bi-directional gated recurrent units, each of which propagates over the characters or words, whereas the simple model deploys an encoder consisting of a two-layer bi-directional gated recurrent unit propagating over the subword units in the source sentence. Both models deploy a two-layer decoder based on the stacked gated recurrent unit architecture [6], an embedding and hidden unit size of 512, and a model vocabulary of 30,000 units. The compositional model uses a trigram vocabulary of the same size, whereas the words in the source and target sides are segmented using linguistically-motivated vocabulary reduction with a target lexicon size of 30,000 units. The models are trained using the Adam optimizer with an initial learning rate of 0.0002 and default values for the other hyper-parameters. We clip the gradient norm at 1.0 [77] and set the dropout to 0.1 after hyper-parameter tuning.

The performance of neural machine translation models in translating Turkish and Italian in the multi-domain case using different types of encoder input representations can be seen in Table 6.3. The better performance of the compositional model in translating Turkish suggests that our approach is beneficial in eliminating the morphological errors caused by segmentation. In Italian, which represents the case of the lowest level of

Language Direction	Model	BLEU	chrF3
Italian-English	Simple (BPE)	**29.02**	**0.5328**
	Compositional	28.66	0.5293
Turkish-English	Simple (LMVR)	23.02	0.4613
	Compositional	**23.13**	**0.4703**

Table 6.3: Experiment results in multi-domain settings. Best scores are in bold font. All improvements over the baseline are statistically significant ($p - value < 0.05$).

lexical sparsity, the source word vocabulary is around 150,000 words in a corpus of approximately 1 million tokens. The higher overall performance of the simple neural machine translation model suggests that subword segmentation may be sufficient in efficiently reducing this vocabulary to fit into a space of 30,000 units.

6.3.3 Accuracy in Translating Rare and Unseen Words

Since the performance of the simple and compositional models have been found to provide comparable in the overall translation accuracy, in order to better illustrate their difference we perform an additional analysis evaluating both models in terms of their actual performance in translating rare and unseen words. This analysis is performed by sampling from the test sets only the sentences that contain singletons (*i.e.* words that are observed once in the training corpus) in the source side or out-of-vocabulary words, and evaluating the translation accuracy obtained with each neural machine translation model on these sentences. This sampling results in 190 sentences in Italian and 470 sentences in Turkish for singletons, and 443 Italian and 1096 Turkish sentences for out-of-vocabulary words. The evaluation of each model on these test sets, results of which are also given in Table 6.4, show that the compositional model is especially advantageous in

translating rare and unseen words and can provide a higher generalization capability to the neural machine translation model.

The results of this analysis, as given in Table 6.4, show that the compositional model translates sentences containing rare words more accurately than the simple model in both languages, where the improvements are **0.53** BLEU points in Italian and **1.22** BLEU points in Turkish. The improvement obtained also in the Italian-to-English translation direction shows that although in overall subword segmentation achieves higher output accuracy, it is still not as efficient as our approach in translating the small portion of rare words in the Italian corpus. Similarly, also in the out-of-vocabulary evaluation, the compositional model again outperforms the simple model by **1.75** BLEU points in Italian and **1.19** BLEU points in Turkish. These findings suggest that the compositional neural machine translation model provides a better approach to representing unseen or rare words, thus, a higher generalization capability compared to conventional approaches to open vocabulary neural machine translation.

Language Direction	**Model**	**BLEU** (Singletons)	**BLEU** (OOVs)
Italian-English	Simple (BPE)	23.54	23.23
	Compositional	**24.07**	**24.98**
Turkish-English	Simple (LMVR)	19.69	20.31
	Compositional	**20.91**	**21.50**

Table 6.4: Translation accuracy of neural machine translation models evaluated only on sentences containing singletons and out-of-vocabulary words. Best scores are in bold font. All improvements over the baseline are statistically significant ($p - value < 0.05$).

6.4 Conclusion

This chapter addressed the problem of translating infrequent words in neural machine translation with the approach of using input representations learned compositionally from character n-grams to operate the neural machine translation model at the level of words, without the requirement of managing a lexical vocabulary. We tested our approach by augmenting the encoder of the neural machine translation model with a bi-directional neural network and evaluating it in the machine translation task from five morphologically-rich languages into English. Our approach obtained significant improvements over the approach of performing neural machine translation with subword or character embeddings, showing promising application for neural machine translation of low-resource and morphologically-rich languages. In the next chapter, we explore the benefit of using compositional input representations also in the target side of the neural machine translation model through a novel hierarchical decoding architecture.

Chapter 7

Hiearchical Character Decoding

7.1 Introduction

In this chapter, we propose a novel decoding architecture which deploys compositional word representations in the decoder of the neural machine translation model, allowing to achieve a more efficient solution to open-vocabulary neural machine translation while translating into morphologically-rich languages. Our decoding architecture processes target sentences at multi-level dynamic time steps, integrating a notion of hierarchy into the decoder, where all word representations are learned compositionally from character embeddings, and perform translation by generating each word character by character based on the predicted word representation. In addition to the decoding architecture, we also present a hierarchical beam search algorithm that allows to take advantage of the hierarchical structure of the decoder. We explore the optimal decoding architecture and target vocabulary units which can aid in minimizing the sparsity and learning reliable representations in the target language in the most computationally efficient manner through an evaluation comparing our approach against subword or character-level decoding in neural machine translation.

7.2 The Method

7.2.1 Model

Similar to the encoder of the compositional neural machine translation model, the hierarchical decoding model has an input layer composed of an embedding layer which encodes embeddings of the characters inside each word in the target sentence, and a character-level bi-directional recurrent neural network which estimates a composition function over the character embeddings to compute the target word representations. The prediction of the each target word is accomplished as in the standard architecture us-

ing the compositional target word representations, which are computed as given in Section 6.2, where the attention mechanism and the target context is predicted by a unidirectional recurrent neural network that operates at the level of words. Instead of classifying the predicted target word in the vocabulary, in the hierarchical decoding model, its distributed representation $\bar{h}_j$ is fed to a character-level unidirectional recurrent neural network to generate the target word character by character by modeling the probability of observing the l_{th} character in the j_{th} word, $p(y_{j,l}|y_{j,<l}, \bar{h}_j)$, given the word prediction and the previous characters in the word.

Similar to Luong and Manning [66], we initialize the character-level recurrent neural network with the attentional vector $\bar{h}_t$, which is computed once for each word in the target sentence based on the word-level context and previously generated words as follows:

$$\bar{h}_j = \tanh(W[c_t; h_j]) \tag{7.1}$$

where $\overline{h}_j$ is the hidden state of the word-level recurrent neural network representing the current target context.

Hierarchical decoding consecutively iterates over the words and characters of the target sentence, therefore each recurrent neural network is updated in dynamic steps based on the word boundaries.

7.2.2 Predictions

In order to achieve efficient decoding with the hierarchical neural machine translation decoder, we implement a **hierarchical beam search** algorithm, suggested also by Ling *et al.* [63]. The algorithm starts decoding by predicting the B most likely characters and storing them in a character beam along with their probabilities. Different than the standard algorithm, the beams are reset each time the generation of a word is complete and the B most likely words computed after the beam search are stored in an

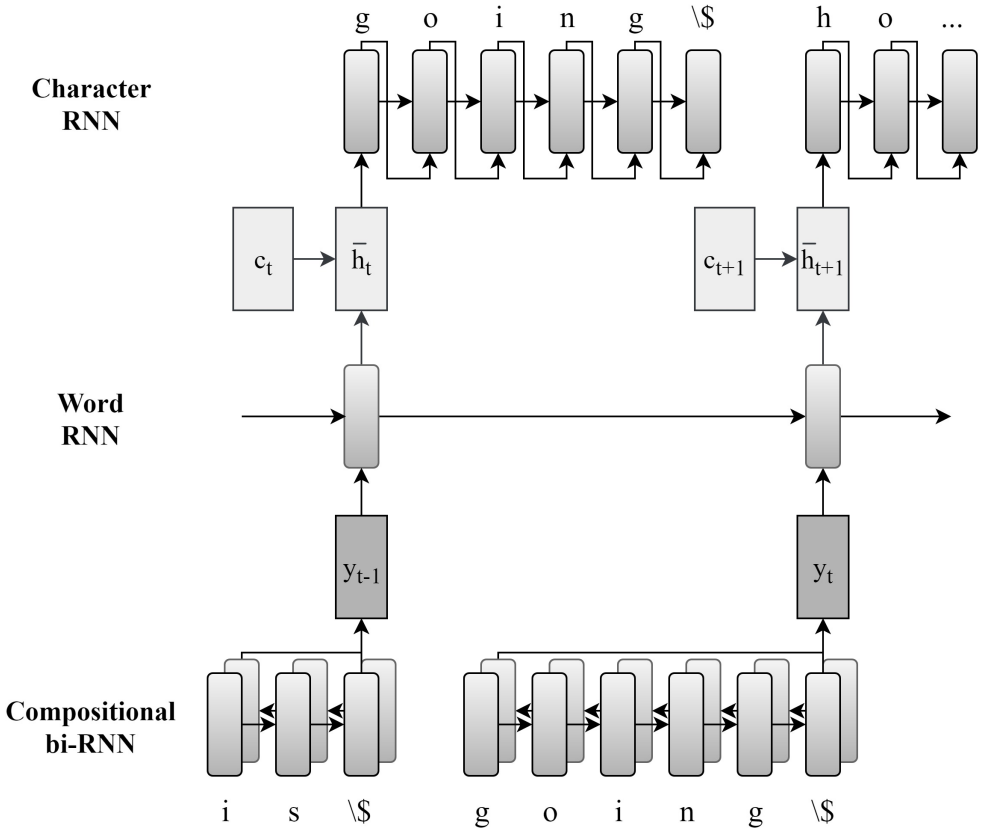


Figure 7.1: Hierarchical character-level decoder: input words are encoded as character sequences and output words are generated as character sequences.

intermediate word-level beam. The word beam is used to compute the B distributed representations corresponding to the most likely B next target words, which are fed to the character-level recurrent neural network to continue the beam search. When the beam search is complete, the most likely character sequence is generated as the best hypothesis. The pseudo-code of the method can be seen below.

7.3 Evaluation

The evaluation of our architecture includes an analysis to determine the set of vocabulary units to use in decoding, followed by our standard evalu-

```
function HierarhicalBeamSearch(Hyp,Best,t)
NewHyp ← ()
for all (seq,score,state) in Hyp do:
  (chars,logpr,stâte) ← CharRNN_Fwd(tail(seq), state)
  for all (c,lp) in (characters,logpr) do:
   hyp=[append(seq,c),score+lp,stâte]
   if (IsSolution(hyp) and
   hyp.score > Best.score)
   then Best=hyp
   else Push(NewHyp,hyp)
  NewHyp ← Prune(NewHyp,Best)
NewHyp ← TopB(NewHyp)
NewHyp.state ← WordRNN_Fwd(NewHyp)
if (NewHyp)
  return BeamSearch(NewHyp,Best,t+1)
else return Best
```

Algorithm 2: The hierarchical beam search algorithm.

ation task of machine translation using the common evaluation benchmark under low-resource and mid-resource settings. The next sections present the findings of different sets of experiments where we compare each open-vocabulary neural machine translation method performing decoding at the level of character, subwords, or hierarchical word-character units.

7.3.1 The Effect of Output Units

We start our study with an analysis of the optimal set of output units that can minimize the search space in decoding by measuring the reconstruction accuracy of words with an auto-encoder based on recurrent neural networks. The input of the auto-encoder is the compositional word representation computed from the character n-grams inside the word with a bi-directional recurrent neural network, as described in Section 6.2. The decoder, a unidirectional recurrent neural network then tries to reconstruct

the given word by predicting the sequence of character n-grams that produced the word.

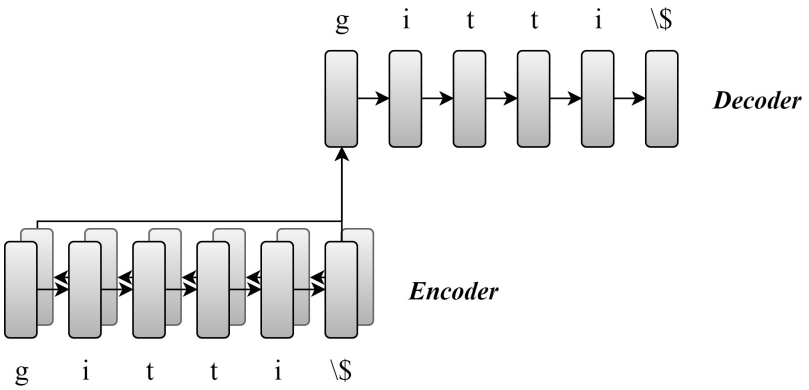


Figure 7.2: The auto-encoder used in the output unit analysis

In our analysis, we use the Turkish language as it has the most complex morphology, and transform the corpus from the TED Talks benchmark where the input and outputs are character n-gram sequences constructing each unique word in the corpus. By using different compositional units such as character unigrams, bigrams or trigrams, we measure the accuracy in reconstructing each word in terms of the percentage of correctly generated character n-grams.

Output Unit	**Reconstruction Accuracy (%)**
Character unigrams	**99.80**
Character bigrams	98.87
Character trigrams	98.21

Table 7.1: Reconstruction accuracy of the character-level auto-encoder

The results of our analysis, as shown in Table 7.1, indicates that a lower level of sparsity in the output units allows to obtain a higher reconstruction accuracy. In light of these findings, we proceed to construct the target vocabulary in our neural machine translation model from the set of characters in the target language.

7.3.2 Effects of Morphological Typology

We evaluate decoding architectures deploying different levels of granularity in the vocabulary units and the attention mechanism, including the standard decoding architecture deployed either with subword [94] or fully character-level [15] units, which constitute the baseline approaches, and the hierarchical decoding architecture, by implementing all in Pytorch [77] within the OpenNMT-py framework [56] in the machine translation task from English into the five morphologically-rich languages included in the TED Talks benchmark.

All models are implemented using gated recurrent units [16] and evaluated while using a comparable number of gated recurrent unit parameters. We use a hierarchical decoding model which deploys 3 layers of gated recurrent units. In order to show the difference of maintaining the context and the attention mechanism at the level of words, we compare our model against a fully character-level decoder which also uses a 3-layer stacked gated recurrent unit architecture. The subword-level decoder has a 2-layer stacked gated recurrent unit architecture, to account also for the larger number of embedding parameters. The models using the standard architecture have the attention mechanism after the first gated recurrent unit layer, and have residual connections after the second layer [6]. The hierarchical decoder implements the attention mechanism after the second layer in order to compute the context vector at the level of words.

The source sides of the training data used in all models, and the target sides of the models using subword embeddings are segmented using byte-pair encoding in German, Czech and Italian and with linguistically-motivated vocabulary reduction in Arabic and Turkish with 16,000 shared merging rules. The small vocabulary size is chosen to minimize the model parameters while keeping the output accuracy at an acceptable level. The

character-level decoders, including the fully character-level model and the hierarchical model, both use a target vocabulary of 150 units. All models use an embedding and hidden unit size of 512. We train the subword-based models using the Adam optimizer with a learning rate of 0.0003, and the character-based models with an initial learning rate of 0.0004 and a decay of 0.9. We use a batch size of 64 and tune the dropout rates as 0.1 for the character-level, 0.2 for the hierarchical, and 0.3 for the subword-level neural machine translation models.

The results of the experiments given in Table 7.2 show that the hierarchical decoder can reach comparable or better performance to the neural machine translation model based on subword units in all languages while using approximately five times less parameters. The improvements are especially evident in Arabic and Turkish, where the hierarchical decoder reaches **1.05** and **0.71** BLEU points of improvement, respectively. In Czech, Italian and German, which constitute the fusional languages, the performance of the two decoders are generally comparable, where in Czech the hierarchical model outperforms the subword unit based model with **0.19** BLEU and in Italian by **0.41** BLEU points. In German, the superior performance of the subword based model suggests the convenience of byte-pair encoding as an open-vocabulary neural machine translation solution.

The character-level neural machine translation model, on the other hand, significantly outperforms the hierarchical model in Turkish by **0.91** BLEU and in Czech by **0.15** BLEU points. As can be seen in the statistical characteristics of the training sets given in Table 4.2, these two directions constitute the most sparse settings, where the training data is quite small, ranging from around 118 to 137 thousand sentences, in addition to large word vocabularies. The improvements are proportional to the amount of sparsity in two languages, as given in the token-to-type rations in the

Model	En-Ar		En-Cs		En-De	
	BLEU	**chrF3**	**BLEU**	**chrF3**	**BLEU**	**chrF3**
Subwords	14.50	0.3992	16.60	**0.4123**	**24.29**	**0.5129**
Characters	12.72	0.3804	**16.94**	0.4103	22.23	0.4884
Hierarchical	**15.55**	**0.4154**	16.79	0.4068	23.91	0.4956

Model	En-It		En-Tr	
	BLEU	**chrF3**	**BLEU**	**chrF3**
Subwords	26.23	**0.5168**	9.03	0.3804
Characters	24.33	0.5007	**10.63**	**0.3810**
Hierarchical	**26.64**	0.5046	9.74	0.3771

Table 7.2: Experiment Results in TED Talks benchmark. Best scores for each translation direction are in bold font. All improvements over the baselines are statistically significant ($p - value < 0.05$).

training corpora (Figure 5.1a), where Turkish has the highest amount of sparsity in the benchmark, followed by Czech as the second most complex language. In cases of high lexical sparsity, learning to translate based on representations of characters might allow to reduce contextual sparsity, leading in better distributed representations of translation units. As the sparsity in the language decreases, either in the form of the training data size, or the morphological complexity of the target language, the advantage of using character-level neural machine translation becomes less evident. In Arabic and Italian, where the training data is almost twice as large as the other languages, using the hierarchical model provides improvements of **2.83** and **2.31** BLEU points, respectively.

7.3.3 The Effect of Data Size

In order to see how the performance of each method scales with increasing data size and multi-domain conditions, we repeat the experiments in the English-to-Turkish and English-to-Italian directions, using the data sets

described in 4.3. The results of these experiments can be seen in Table 7.3.

Model	En-It		En-Tr	
	BLEU	**chrF3**	**BLEU**	**chrF3**
Subwords	26.66	**0.5299**	**10.65**	**0.3893**
Characters	18.01	0.4454	8.94	0.3274
Hierarchical	**27.28**	0.4830	10.35	0.3870

Table 7.3: Experiment results in multi-domain settings. Best scores for each translation direction are in bold font. All improvements over the baselines are statistically significant ($p - value < 0.05$).

In the English-to-Italian direction, an increase in the data size leads to a slight increase in the accuracy of the hierarchical and subword-based NMT decoders, where the hierarchical model outperforms the subword-based one by **0.66** BLEU points. The fully character-level model obtains significantly lower accuracy compared to both approaches, where its accuracy on the same test set drops by **6.32** BLEU points, which might be due to the increased level of ambiguity in character embeddings when the model is trained over the mid-resource noisy corpora without increasing the network capacity. Studies have shown that fully character-level models could potentially reach the same performance with the subword-based NMT models by increasing the amount of capacity [15]. However, the consistency of the improvements obtained with the hierarchical decoding architecture under different settings, along with its lower average convergence time, suggests the advantage of using explicit modeling of word boundaries in achieving a more computationally efficient solution to character-level translation.

Increased data size in the English-to-Turkish translation direction, on the other hand, does not necessarily yield a reduction in lexical sparsity, on the contrary, increases the possibility of observing even more rare words, either in the form of morphological inflections due to the complex agglu-

tinative morphology of Turkish, or ambiguous terminology raising from the multi-domain characteristics. In this experiment, the character-level model again experiences a drop in performance, whereas the subword-based model achieves the best accuracy by **0.30** BLEU points above the hierarchical model, suggesting a better capacity in reducing the lexical sparsity.

7.4 Conclusion

The perception and generation of language are fundamentally different processes, and require adopting different approaches to their modeling. The results of Chapter 6 have shown the advantage of performing translation at the level of words by using compositional word representations. In the case of encoding, adding phonetic context into the vocabulary units that compose the words by means of intersecting character trigrams allowed learning richer representations which are especially useful in representing rare or unseen words.

In generation, on the other hand, the level of sparsity in the vocabulary units is crucial as it defines the granularity of the search space. Our analysis showed that the highest reconstruction accuracy for a given word is obtained using a vocabulary of interior units with the lowest level of granularity, which is the set of characters in the target language. However, in the task of generating sentences, a higher level of granularity, such as performing the search over the sets of all possible words, rather than subwords or characters in the sentence, could be beneficial, as the uncertainty at each step in the generation process would be reduced. This is due to the fact that for each correct word in the target sentence, there might be multiple possible sets of subword units that have potentially similar likelihoods to the likelihood of generating the correct word.

This chapter explored the idea of performing the decoding procedure in neural machine translation in a multi-dimensional search space in terms of asynchronous word and character level generation. The hierarchical decoder we proposed achieved better performance than the conventional open-vocabulary neural machine translation solutions based on subword units in multiple languages while using a significantly less number of parameters. The main advantage of the hierarchical decoding model is that

it learns to translate based on word representations learned by the lexical context, which is linguistically more coherent, as semantics and syntax of the language are typically defined at the lexical level. On the other hand, by eliminating the necessity to store any of the word representations, it can drastically reduce the computational cost of the neural machine translation model.

On the other hand, the requirement of observing words a sufficient number of times in different lexical contexts in order to model the lexical distribution efficiently, and learn a reliable morphological composition function such that the model can extract generalizable patterns, is an important limitation for its usage under low-resource settings. In the next chapter, we present an approach for modeling this variance in the lexical distribution and aiding the model in learning more reliable lexical representations under conditions of high lexical sparsity.

Chapter 8

Decoding with a Stochastic Morphology Model

8.1 Introduction

As described in Chapter 2, neural machine translation models are conventionally trained based on the approach of maximizing the log-likelihood on a training corpus in order to predict the model parameters θ. In this approach, belonging to the school often referred as frequentist statistical modeling, one generally assumes that the data is dynamic, and the model parameters, on the other hand, are static variables, and are learned such that they capture the probabilistic model represented for that data. Once the parameters are learned, their values are deterministic and are used to make predictions on new data. The main problem with this approach is that the uncertainty in model predictions is often ignored, which may be crucial in cases where the data has high variance, such that there might be many likely hypotheses in a given context. In order to induce an uncertainty in the model outcomes, one can instead opt for the Bayesian modeling approach, where the model is also probabilistic, including its parameters and its possible predictions. In this approach, parameters are random variables, the prior distribution of which can be defined a-priori, which aids the learning process as instead of learning the parameter values, the model needs to infer a set of stochastic latent variables, which can represent the probabilistic model, based on the posterior distribution obtained from the data. Each model parameter, and consequently predictions, would then always contain a level of uncertainty, essentially allowing one to have a conceptually more inclusive probabilistic model which can, in principle, have a better notion on generalization over variance in the data and model outcomes.

Bayesian modeling is relevant to our study since it can reduce the reliance on data for learning a given task, aiding in learning statistical models under high amount of data sparsity. In this chapter, we study the benefit

of applying the approach of Bayesian modeling in order to learn the distributed representations of words in the neural machine translation model. For this purpose, we formulate a novel stochastic model of morphology, the variables of which represent the distribution of morphological features inside words, and are learned such that they can be shared to represent different words, aiding in generalizing the contextual representations of words across their different surface forms. In this model, instead of the lexical representations learned by the word-level recurrent neural network, the predictions with the hierarchical decoder are done based on word representations encoded by combining a set of latent morphological features, which allows to model the uncertainty in the target words and their forms. We present the design of our model, some related work which have also adopted stochastic modeling in natural language processing tasks, such as morphological reinflection [112] or sentiment classification [7], and the updated formulations of the objective function used to infer the latent variables during training. We evaluate our method in the three most sparse languages in our evaluation benchmark that were found to be problematic with the hierarchical decoding model in Chapter 7, and a detailed analysis comparing the generative properties of different methods for open-vocabulary neural machine translation.

8.2 The Method

8.2.1 Model

Our generative latent morphology model for neural machine translation formulates word formation in terms of a stochastic process, which induces a stronger bias towards achieving a data-driven model of morphological analysis. In our stochastic morphology model, each word is represented using two latent variables corresponding to its major morphological com-

ponents: a continuous vector aimed at representing the lexical semantics of the word, or its *lemma*, and a set of approximately discrete sparse features aimed at capturing the word's *inflectional features*.

Using a latent variable model for this formulation has mainly two advantages. First, deterministic models are by definition unimodal: when presented with the same input, or the context, they always produce the same output. When we model a sequential generation process, it is reasonable to expect a large degree of uncertainty since most of the context essential for generating the correct output may not yet be generated. For instance, while generating a word starting with a noun root which should receive a prefix before the generation of the root, we may continue by inflecting the word differently depending on the latent mode of operation we are at, such as nominative, accusative or dative noun. Second, in latent variable models, a-priori choice of the statistical distribution provides a mechanism to favour a particular type of representation, aiding the model in capturing the underlying corpus statistics more reliably in case of data sparseness. In our case, we use **sparse** distributions for inflectional features to accommodate the fact that morphosyntactic features are discrete in nature.

Our latent variable model is an instance of variational auto-encoders [54]. A variational auto-encoder consists of an encoder, also called as an **inference model**, which learns to encode a representation of the input data into a parameter vector representing a stochastic variable with a predefined statistical distribution, and a decoder, also called as the **generative model**, which aims to reconstruct the original input with minimal loss by predicting a sample from the learned probability distribution during test time. Being widely used in unsupervised generative models, variational auto-encoders provide an efficient means to encode an input in terms of some latent feature representations, which can also be regarded as a com-

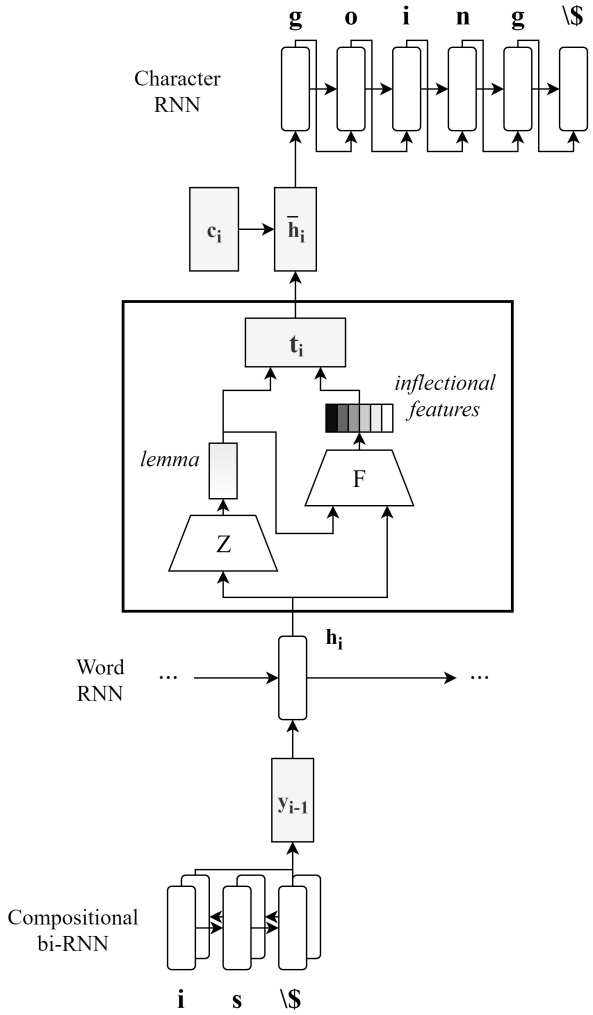


Figure 8.1: The latent morphology model for computing word representations

pression mechanism.

Similar to previous approaches to unsupervised morphology learning based on the minimum-description length principle [89, 39, 72, 26], we model the morphology learning process as a compression task using two variational auto-encoders, one for the estimation of a lexicon (*i.e.* clustering of a set of lexemes) and another one for inferring the set of inflectional features, although the inference of both variables are accomplished in vector space, based on the distributed word representations in the hierarchical target language model, which based on strong evidence from many studies

[107, 85], have shown to contain many cues on the phonetic and semantic features of words.

During decoding, we first sample a Gaussian-distributed representation in context, representing the **lemma**, inspired by the model of Zhou and Neubig [112] for morphological reinflection.[1]

$$\begin{aligned} Z_i|x, y_{<i} &\sim \mathcal{N}(\mathbf{u}_i, \mathrm{diag}(\mathbf{s}_i \odot \mathbf{s}_i)) \\ u_i &= W_u h_i + b_u \\ s_i &= \mathrm{softplus}(W_s h_i + b_s) \end{aligned} \tag{8.1}$$

where two single-layer perceptrons, one of which predicts the location vector in $\mathbb{R}^d$ with parameters W_u and b_u, and another one the scale vector in $\mathbb{R}^d_{>0}$ with parameters W_s and b_s, of the Gaussian variable, from the word-level decoder hidden state $\mathbf{h}_i$, which essentially represents the input x and the target history $y_{<i}$. Note that the softplus activation allows converting all variance values to positive and non-zero.

In practice, we sample z_i via a reparameterization in terms of a fixed Gaussian, namely,

$$z_i = \mathbf{u}_i + \epsilon_i \odot \mathbf{s}_i \tag{8.2}$$

for $\epsilon_i \sim \mathcal{N}(0, I_d)$. This is known as the **reparameterization trick** [54], which allows back-propagation through stochastic units [81].

Second, we sample a vector f_i of K sparse scalar **inflectional features** conditioned on the source x, the target prefix $y_{<i}$, and the sampled lemma z_i. We model sampling of f_i conditioned on z_i in order to capture the insight that inflectional transformations typically depend on the category of a lemma [52].

[1] **Notation** We use capital Roman letters for random variables (and lowercase letters for assignments). Boldface Roman letters are reserved for neural network output vectors, and $\odot$ stands for elementwise multiplication.

Having sampled f_i and z_i, the representation of the ith target word is computed as a linear combination of z_i and f_i,

$$\hat{w}_j = \mathbf{W}_c[z_j, f_j] + \mathbf{b_c} \tag{8.3}$$

where $\mathbf{W_c}$ and $\mathbf{b_c}$ are learnable weight and bias parameters.

As shown in Figure 8.1, our model generates each word character by character auto-regressively by conditioning on the word representation $\mathbf{t}_i$ predicted by the latent morphology model, the current context $\mathbf{c}_i$, and the previously generated characters following the same hierarchical decoding structure described in Section 7.2.

8.2.2 Sparse Features

Since each target word y_i may have multiple inflectional features, ideally, we would like f_i to be K feature indicators, which could be achieved by sampling from K independent Bernoulli distributions parameterized in context. The problem with this approach is that sampling Bernoulli outcomes is non-differentiable, thus, their training requires gradient estimation [110] and sophisticated variance reduction techniques.

Instead, Louizos *et al.* [65] proposed obtaining a continuous sample $c \in (0, 1)$ from a distribution for which a reparameterization exists and stretch it to a continuous support $(l, r) \supset (0, 1)$ using a simple linear transformation

$$s = l + (r - l)c \tag{8.4}$$

where $c \in (0, 1)$ is the original continuous variable. A rectifier is then employed to map the negative outcomes to 0 and the positive outcomes larger than one to 1, *i.e.* $f = \min(1, \max(0, s))$. The rectifier is only non-differentiable at $s = 0$ and at $s = 1$, however, because the stretched variable s is sampled from a continuous distribution, the chance of sampling $s = 0$ and $s = 1$ is essentially 0.

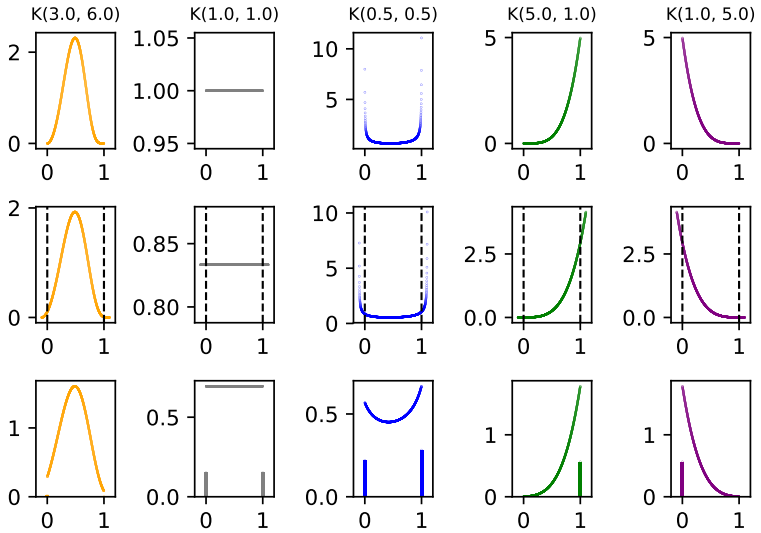


Figure 8.2: The top row shows the density function of the continuous base distribution over $(0, 1)$. The middle row shows the result of stretching it to include 0 and 1 in its support. The bottom row shows the result of rectification: probability mass under $(l, 0)$ collapses to 0 and probability mass under $(1, r)$ collapses to 1, which cause sparse outcomes to have non-zero mass. Varying the shape parameters (a, b) of the underlying continuous distribution changes how much mass concentrates outside the support $(0, 1)$ in the stretched density, and hence the probability of sampling sparse outcomes.

This stretched-and-rectified distribution allows the sampling procedure to become differentiable with respect to the parameters of the distribution, to sample sparse outcomes with an unbiased estimator, and and to calculate the probability of sampling $f = 0$ and $f = 1$ in closed form as a function of the parameters of the underlying distribution, which corresponds to the probability of sampling $s < 0$ and $s > 1$, respectively. Bastings *et al.* [7] proposed using this method with latent variables based on the Kumaraswamy distribution [59], a two-parameters distribution that closely resembles a Beta distribution and is sparse whenever its (strictly positive) parameters are between 0 and 1.

For each token y_i, we sample K independent Kumaraswamy variables

in context,

$$\begin{aligned} C_{i,k}|x, y_{<i}, z_i &\sim \text{Kuma}(a_{i,k}, b_{i,k}) \quad k = 1, \ldots, K \\ [\mathbf{a}_i, \mathbf{b}_i] &= \text{softplus}(W_f([z_i, \mathbf{h}_i]) + b_f) \end{aligned} \tag{8.5}$$

where W_f and b_f are the parameters of the single-layer neural network predicting the parameters of the Kumaraswamy variable, a and b.

This sampling results in a continuous random vector c_i in the support $(0, 1)^K$.[2] We then stretch-and-rectify the samples via $f_{i,k} = \min(1, \max(0, l - (r - l)c_{i,k}))$ making f_i a random vector in the support $[0, 1]^K$.[3]

The probability that $f_{i,k}$ is exactly 0 is

$$\pi_{i,k}^{\{0\}} = \int_0^{\frac{-l}{r-l}} \text{Kuma}(c|a_{i,k}, b_{i,k})c \tag{8.6a}$$

and the probability that $f_{i,k}$ is exactly 1 is

$$\pi_{i,k}^{\{1\}} = 1 - \int_0^{\frac{1-l}{r-l}} \text{Kuma}(c|a_{i,k}, b_{i,k})c \tag{8.6b}$$

and therefore the complement

$$\pi_{i,k}^{(0,1)} = 1 - \pi_{i,k}^{\{0\}} - \pi_{i,k}^{\{1\}} \tag{8.6c}$$

is the probability that $f_{i,k}$ be any continuous value in the open set $(0, 1)$.

In Section 8.2.4, we will derive regularizers based on $\pi_{i,k}^{(0,1)}$ to promote sparse outcomes to be sampled with a large probability.

8.2.3 Parameter estimation[4]

Parameter estimation of neural network models is typically done via maximum-likelihood estimation, where we approach a local minimum of the negative

[2] In practice we sample $c_{i,k}$ via a reparameterization of a fixed uniform variable, namely, $c_{i,k} = (1 - (1 - \varepsilon_{i,k})^{1b_{i,k}})^{1a_{i,k}}$ where $\varepsilon_{i,k} \sim \mathcal{U}(0, 1)$, which much like the Gaussian reparameterization enables back-propagation through samples [71].

[3] We use $l = -0.1$ and $r = 1.1$. Figure 8.2 illustrates different instances of this distribution.

[4] The formulations in this section were done in collaboration with Wilker Aziz from the University of Amsterdam.

log-likelihood function via stochastic gradient descent with gradient computation automated by the back-propagation algorithm. Unfortunately, maximum-likelihood estimation for variational auto-encoders is intractable because the marginal likelihood requires summing over an infinite number of configurations of the lemma variables z_1^l as well as of the morphological features f_1^l. To circumvent this intractability, we employ variational inference [50].

Using the following shorthand notation:

$$\alpha(z_i) \triangleq p(z_i|x, y_{<i}, z_{<i}, f_{<i}, \theta) \tag{8.7a}$$

$$\beta(f_i) \triangleq \prod_{k=1}^{K} p(f_{i,k}|x, y_{<i}, z_{<i}, f_{<i}, z_i, \theta) \tag{8.7b}$$

$$\gamma(y_i) \triangleq \prod_{j=1}^{l_i} p(y_{i,j}|x, y_{<i}, z_{\leqslant i}, f_{\leqslant i}, y_{i,<j}, \theta) \ . \tag{8.7c}$$

The log-likelihood for a single data point can be formulated as:

$$\log p(y|x, \theta) = \log \int \prod_{i=1}^{l} \alpha(z_i)\beta(f_i)\gamma(y_i) z f \tag{8.8}$$

Since the above computation is intractable, we employ variational inference [50], where we optimize a lower-bound on the log-likelihood

$$\mathbb{E}_{q(z,f|x,y,\lambda)} \left[\sum_{i=1}^{l} \log \frac{\alpha(z_i)\beta(f_i)\gamma(y_i)}{q(z, f|x, \lambda)} \right] \tag{8.9}$$

expressed with respect to an independently parameterized posterior approximation $q(z, f|x, y, \lambda)$. For as long as sampling from the posterior is tractable and can be performed via a reparameterization, we can rely on stochastic gradient-based optimization. In order to have a compact parameterization, we choose

$$q(z, f|x, y, \lambda) := \prod_{i=1}^{l} \alpha(z_i)\beta(f_i) \ . \tag{8.10}$$

This simplifies the lowerbound, which then takes the form of l nested expectations, the ith of which is $\mathbb{E}_{\alpha(z_i)\beta(f_i)}[\log \gamma(y_i)]$. In contrast to the general approach, we do not deploy an additional inference model, thus, our approximate posterior is in fact, also our parameterized prior.

Concretely, for a given source sentence x, target prefix $y_{<i}$, and a latent sample $z_{\leqslant i}, f_{\leqslant i}$, we obtain a single-sample estimate of the loss by computing $\mathcal{L}_i(\theta) = -\log \gamma(y_i)$. Although this objective does not particularly promote sparsity, we employ sparsity-inducing regularization techniques that will be discussed in the next section.

8.2.4 Regularization

In order to promote sparse distributions for the inflectional features, we apply a regularizer inspired by expected L_0 regularization [65]. Whereas L_0 is a penalty based on the number of non-zero outcomes, we design a penalty based on the expected number of continuous outcomes, which corresponds to $\pi_{i,k}^{(0,1)}$ as shown in Equation (8.6). For a given source sentence x, target prefix $y_{<i}$, and a latent sample $z_{<i}, f_{<i}$, we aggregate this penalty for each feature

$$\mathcal{R}_i(\theta) = \sum_{k=1}^{K} \pi_{i,k}^{(0,1)} \tag{8.11}$$

and add it to the cost function with a positive weight ρ. The final loss of the neural machine translation model is

$$\mathcal{L}(\theta|\mathcal{D}) = \sum_{x,y\sim\mathcal{D}} \sum_{i=1}^{|y|} \mathcal{L}_i(\theta) + \rho\mathcal{R}_i(\theta) \ . \tag{8.12}$$

8.2.5 Predictions

In our model, obtaining the conditional likelihood for predicting the most likely hypothesis requires marginalization of the latent variables, which is

intractable. An alternative approach is to heuristically search through the joint distribution,

$$\arg\max_{y,z,f} \; p(y, z, f|x) \tag{8.13}$$

rather than the marginal, an approximation that has been referred to as *Viterbi decoding* [97]. During beam search, we populate the beam with alternative target words, and for each prefix $y_{<i}$ in the beam, we resort to deterministically choosing the latent variables based on a single sample which we deem representative of their distributions, which is a common heuristic in previous work which used variational auto-encoding in machine translation [111, 90]. For unimodal distributions, such as the Gaussian $p(z_i|x, y_{<i}, z_{<i}, f_{<i})$, we use the analytic mean, whereas for multimodal distributions, such as the Hard Kumaraswamy $p(f_i|x, y_{<i}, z_{\leqslant i}, f_{<i})$, we use the argmax. The beam search is implemented with the hierarchical search algorithm described in Section 7.2.2.

8.3 Evaluation

We evaluate our model by comparing its machine translation accuracy against three baselines used in the previous chapter, including the standard architecture learning translation at the level of subword units (segmented with byte-pair encoding or linguistically-motivated vocabulary reduction) or characters, and the hierarchical decoding architecture employed for generating all words in the output character by character. For comparison, we use the results of the experiments given in Section 7.3, where the latent morphology model is also implemented using Pytorch [77] within the OpenNMT-py framework [56]. In order to see the performance of our model in languages with different morphological typology, we model the machine translation task from English into three most sparse languages in our benchmark: Arabic, Czech and Turkish. The latent morphology model

Model	AR		CS		TR	
	BLEU	**chrF3**	**BLEU**	**chrF3**	**BLEU**	**chrF3**
Subwords	14.50	0.3992	16.60	**0.4123**	9.03	0.3804
Char.s	12.72	0.3804	16.94	0.4103	10.63	0.3810
Hierarch.	15.55	0.4154	16.79	0.4068	9.74	0.3771
Hierarch. with LMM	**16.06**	**0.4251**	**16.97**	0.4095	**10.93**	**0.3889**

Table 8.1: Results of the experiments in Arabic (AR), Czech (CS) and Turkish (TR) under low-resource settings using in-domain training data. All improvements over the baselines are statistically significant ($p - value < 0.05$).

uses the same number of layers as the hierarchical gated recurrent unit based decoder, where the middle layer is augmented using four multi-layer perceptrons with 256 hidden units. We use a lemma vector dimension of 150, 10 inflectional features and set the regularization constant to $\rho = 0.4$. We use the same network hyper-parameters as the baselines described in Section 7.3 during training and decoding.

8.3.1 The Effect of Morphological Typology

The experiment results given in Table 8.1 shows the translation accuracy obtained with each model. Using the latent morphology model provides improvements of **0.51** and **0.30** BLEU points in Arabic and Turkish over the best performing baselines, respectively. The fact that our model can efficiently work in both Arabic and Turkish suggests that it can handle the generation of both concatenative and non-concatenative morphological transformations. The results in the English-to-Czech translation direction suggest that there might not be a specific advantage of using either method for generating fusional morphology, where morphemes are already optimized at the surface level, although our model is still able to achieve translation accuracy comparable to the character-level model.

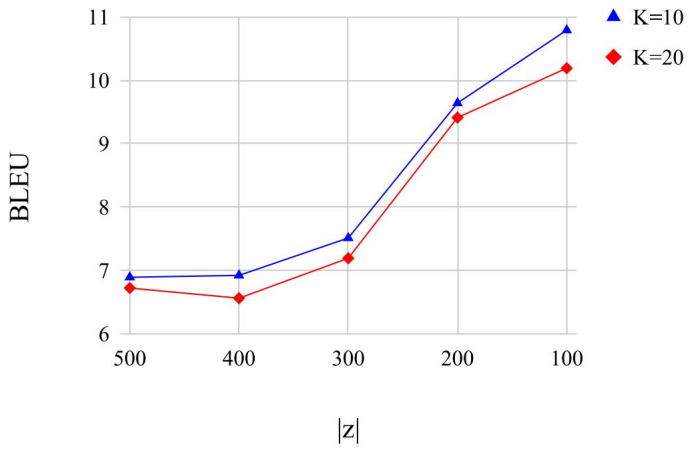


Figure 8.3: The effect of feature dimensions on translation accuracy in Turkish

8.3.2 The Effect of Feature Dimensions

We investigate the optimal lemma and inflectional feature dimensions by measuring the accuracy in English-to-Turkish translation using different feature vector dimensions. The results given in Figure 8.3 show that gradually compressing the word representations computed by the recurrent network hidden states, with an original dimension of 512, from 500 to 100, leads to a consistent increase in the output accuracy, suggesting that encoding more compact representations might provide the model with a better generalization capability. Our results also show that using a feature dimension of 10 is sufficient in reaching the best accuracy.

8.3.3 The Effect of Data Size

The results of the experiments conducted in the English-to-Turkish translation direction by increasing the amount of training data to include corpora from different domains can be seen in Table 8.2. Increased data size in this case does not necessarily yield a reduction in lexical sparsity, on the contrary, increases the possibility of observing even more rare words, either in the form of morphological inflections due to the complex agglu-

Model	TR	
	BLEU	**chrF3**
Subwords	10.65	0.3893
Char.s	8.94	0.3274
Hierarch.	10.35	0.3870
Hierarch. with LMM	**11.48**	**0.3939**

Table 8.2: Results of the experiments in Turkish (TR) under low-resource settings using multi-domain training data. All improvements over the baselines are statistically significant ($p - value < 0.05$).

tinative morphology of Turkish, or ambiguous terminology raising from the multi-domain characteristics. The comparison of the results with the first experiment using single-domain data show that in case of increased sparsity in data our model reaches a larger improvement of **0.82** BLEU points over the best baseline, which is the subword-based neural machine translation model.

8.3.4 Predicting Unseen Words

In order to illustrate how each model performs in predicting unseen words, we perform an analysis by sampling the sentences in the development set which contain out-of-vocabulary words, and compute the average perplexity per character on these sentences using different models, as suggested by Cotterell *et al.* [24]. The analysis is performed using the multi-domain English-to-Turkish neural machine translation models operating on the levels of subwords, characters, or the hierarchical decoding models.

In general, the lowest and highest perplexities are obtained using the character and subword-based models respectively, indicating that increased granularity aids in reducing the uncertainty during prediction, with the exception of Czech, where the values are almost comparable. Due to its stochastic nature, our model yields higher perplexity values compared to

the hierarchical model, whereas the values range between subword and character-based models, possibly finding an optimal level of granularity between the two solutions.

In order to better illustrate the generative properties of different models, we present some sample translations in Tables 8.4 and 8.5, obtained by translating English into Turkish using the neural machine translation models trained on the multi-domain corpus. The input sentences are selected such that they are sufficiently long so that one can see the ability of each model in capturing long-distance grammatical dependencies. In this example, all models fail to predict the correct translation of *'(to) graph'* in the noun phrase *'the measure of the technology that I'm trying to graph'* corresponding to (*'çizmek'*), and assume that the input is a noun, probably because the verbal form of the word has not been observed before. The character-based decoder demonstrates the worst capacity in handling these dependencies within the phrase, accompanied by many other mistakes such as often repeating or missing words in the sentence, by generating a translation to the phrase that literally means *'at the time of the technology which I am trying to make a graph'*. On the other hand, it can generate at least a verbal phrase meaning *'to make a graph'*, which all other models fail at achieving. The subword-level decoder generates an output with meaning *'the technology I work in the graph'*, showing mistakes both at grammatical and lexical level. The output of the hierarchical decoder makes several mis-

Model	**Perplexity**		
	AR	**CS**	**TR**
Subwords	2.84	2.62	2.78
Char.s	2.46	2.61	2.38
Hierarch.	2.59	2.65	2.46
Hierarch. with LMM	2.68	2.71	2.59

Table 8.3: Normalized perplexity measures per characters in different languages

takes in the word order and morphological inflection, although gets very close to the correct translation, with the generated output meaning *'whatever measure of the technology I am working to the graph'*. The stochastic decoder, on the other hand, generates an output that is, while being not in the same phrasing as in the reference sentence, grammatically correct and conveys a similar meaning to the input, with the translation *'the measure of the technology I am working on the graph'*. Only the stochastic decoder inflects the noun with the preposition *to* into the correct form.

Input	well the answer is, if I drew it on a normal curve where, let's say, this is years, this is time of some sort, and this is whatever **measure of the technology that I'm trying to graph**, the graphs look sort of silly.
Output *Subword-based Decoder*	cevap şu, eğer bunu normal bir eğriye çizersem, diyelim ki bu yıllar yıllar, bu bir tür zaman, ve **grafikte çalıştığım teknoloji** her ne ise aptalca gözüküyor.
Output *Character-based Decoder*	cevap bunu normal bir şekilde çekersem, bu tür bir şekilde çekilebilir, ve bu bir şekilde **grafik yapmaya çalıştığım teknolojinin zamanında grafiğin** grafiklerini ölçebilirim.
Output *Hierarchical Decoder*	cevap şu, eğer normal bir eğri çizdiğimde, diyelim ki, bu yıllar, bu bir şekilde bir şey, ve **grafikte çalıştığım teknolojinin ne ölçüsü** her ne ölçüyor, grafikler aptalca.
Output *Hierarchical Stochastic Decoder*	cevap şu ki, eğer bunu normal bir eğriye çevirdim, diyelim ki, bu yıllar, bu bir çeşit zaman, ve **grafiğe çalıştığım teknolojinin ölçüsü**, grafikler aptalca görünüyor.
Reference	cevabı şöyle; eğer bunu normal bir eğri üzerinde çizersem diyelim ki burası yıllar, bir çeşit zaman, ve bu da **teknolojinin çizmeye çalıştığım** hangi **ölçeği** ise o. grafikler biraz saçma görünür.

Table 8.4: Example translations with different approaches in Turkish

The second example requires translating a sentence from a typical con-

versation, which, requires remembering a long context with many references. We highlight the words in each output sequence that is generated for the first time.

Input	when a friend of mine told me that I needed to see this great video about a guy protesting bicycle fines in New York City, I admit I wasn't very interested.
Output *Subword-based Decoder*	**bir arkadaşım New York'ta bisiklet protestosunu** protesto etmek **için bu filmi izlemeye ihtiyacım olduğunu söylemişti.**
Output *Character-based Decoder*	**bana bir arkadaşım** bana **New York'ta bir adam ile ilgili** bir adam hakkında **görmem gereken bir** adam hakkında görmem gerektiğini **söyledi.**
Output *Hierarchical Decoder*	**bir arkadaşım New York'ta bisiklet yapmaya ihtiyacım olduğunu söylediği zaman, kabul ettim**.
Output *Hierarchical Stochastic Decoder*	**bir arkadaşım bana, New York City'teki bisiklet finalleri ile ilgili bir adam görmek istediğimi söyledi, çok ilgili olmadığımı kabul ediyorum.**
Reference	bir arkadaşım New York şehrindeki bisiklet cezalarını protesto eden bir adamın bu harika videosunu izlemem gerektiğini söylediğinde, kabul etmeliyim ki çok da ilgilenmemiştim.

Table 8.5: Example translations with different approaches in Turkish

In this example, most models fail to generate a complete translation, starting to forget the sentence history after the generation of a few words, indicated by the start of generation of repetitions of the previously generated words. The character-level decoder seems to have the shortest memory span, followed by the subword-based decoder, which completely omits the second half of the sentence. The hierarchical decoder generates a full sentence, although it still misses the last four words in the input, and has a few lexical errors. The stochastic decoder is the only model that generates an almost complete translation, but still translates the phrase *'a movie about*

a guy protesting bicycle fines in...' inaccurately, with the output literally meaning *'a man about the bicycle finals in...'*.

8.3.5 Feature Variations

In order to understand if the latent inflectional features in fact capture information about variations related to morphological transformations, we try generating different surface forms of the same lemma by assigning different values to the inflectional features. We use the hierarchical stochastic decoder to translate the English word *'go'*, and after sampling the lemma, we fix its value and vary the values of the inflectional features at random positions for generating different outputs. Table 8.6 presents different sets of feature values and the corresponding outputs generated by the decoder.

Features	**Output**	**English Translation**
[1,1,1,1,1,1,1,1,1,1]	git	go *(informal)*
[0,1,1,1,1,1,1,1,1,1]	'*a* git	*to* go
[0,1,0,1,1,1,1,1,1,1]	'*da* git	*at* go
[0,0,0,1,1,0,0,1,1,0]	gidin	go *(formal)*
[1,1,0,0,0,0,1,0,1,1]	gitmek	to go *(infinitve)*
[0,0,1,0,0,0,0,0,0,1]	gidiyor	(he/she/it is) going
[[0,0,0,0,0,0,0,0,1,0]	gidip	*by* going *(gerund)*
[0,0,1,1,0,0,1,0,1,0]	gidiyoruz	(we are) going

Table 8.6: Outputs of the latent morphology model based on the lemma *'go'* and different sets of inflectional features

The model generates different surface forms for different sets of features, confirming that latent variables encode information related to the infinitive form of the verb, as well as its formality conditions, prepositions, person, number and tense. We also observe that many trials based on different feature combinations may result in the same outputs, although some fea-

ture values may not be set in a single-word context. Varying the features individually does not necessarily yield distinct changes in the output, suggesting that some features may act jointly in determining the word form.

8.4 Conclusion

In this chapter, we presented a stochastic morphology model which formulates the process of word formation through a set of hierarchical latent variables that are learned in a completely unsupervised fashion based on phonetic features of words and their lexical context. We integrated this morphology model within the neural machine translation decoder in order to promote sparsity in lexical representations and induce uncertainty on the surface forms during prediction. Our model significantly improved the translation accuracy for the most sparse languages in our evaluation benchmark, Arabic and Turkish, suggesting that it can learn both templatic and agglutinative morphology under very low-resource settings. We also presented an analysis indicating the improvements obtained with the hierarchical stochastic modeling approach, which suggested that modeling translation based on a context defined at the lexical level aids in learning better grammatical and contextual dependencies. Moreover, the stochasticity integrated in the decoder improves the capacity of the hierarchical decoding model in learning better lexical representations, remembering longer context, and often times being able to generalize to different word forms. Our model demonstrated promising application for improving the machine translation accuracy for morphologically-rich and low-resource languages.

Chapter 9

Conclusion

This dissertation addressed the vocabulary limitation in neural machine translation, specifically the difficulty in learning to represent and generate unknown words, and proposed to solve this task by enhancing the model with the capability of learning morphology in a completely unsupervised fashion. In order to achieve a model which has integral knowledge about the structure of words, we gradually integrated methods for unsupervised learning of morphology into the neural machine translation model, and evaluated different approaches to open-vocabulary neural machine translation in a machine translation benchmark consisting of six languages with distinct morphological typology.

We started our research in Chapter 5 by proposing a novel statistical morphological segmentation algorithm which can be used for constructing a subword-level translation vocabulary. Our experiments confirmed the benefit of preserving morphological information during translation by showing improvements in translation accuracy of three of the languages in our evaluation benchmark. On the other hand, our analysis demonstrated many drawbacks related to the approach of vocabulary reduction based on statistical subword segmentation, the main one being the requirement of tuning many hyper-parameters which are not optimized for the transla-

tion task, and the fact that different algorithms seem to be convenient for different sets of languages with a certain morphological typology.

Due to the limitations of implementing the neural machine translation model based on subword units, we extended our approach in Chapters 6 and 7 to model translation at the level of words by augmenting the input embedding layers of the model with a word composition layer, which learns distributed representations of words from character n-grams. Our approach showed to be a promising method to represent unknown words in various languages with different morphological typology, proposing a more generic solution to overcome the vocabulary limitation in neural machine translatin. However, processing the sentence context at the level of words also demonstrated the requirement of using larger training resources, since there is a higher level of contextual sparsity when the distributed representations are learned at the level of words rather than subwords, especially in morphologically-rich languages.

In order to reduce the reliance of the hierarchical neural machine translation model to high amounts of parallel training resources which are not available in many languages and providing the decoder with better generalization capability, in Chapter 8 we finally proposed a novel variational inference algorithm which is used to encode word representations in terms of latent morphological features which are shared among different words, allowing to explicitly model the uncertainty in the surface forms of words during generation. Our model provided significant improvements in translation accuracy for languages with highly complex morphology even under low-resource settings.

Besides the improvements gained with our models for translating morphologically-rich and low-resource languages, the models we developed also demonstrated to be less demanding than conventional methods in terms of the number of parameters to be learned and stored. On the other hand,

our methods were deployed only with recurrent architectures, which are no longer preferred in neural machine translation due to slow processing and convergence times. Implementing our approaches with the feed-forward architectures used today [108] could allow reaching much efficient models, although we leave this to future work. Another possible direction includes exploring the advantage of finding morphological patterns in multi-lingual neural machine translation, which has shown to be useful for low-resource languages. Languages with similar morphological typology could potentially benefit the transfer of morphological information from the high-resource to the low-resource one if the model is trained for their joint translation. Since our methods are unsupervised statistical methods, they can be used in multi-lingual settings to develop a joint morphology model, aiding the decoder in generating better translations, for instance in a morphologically-complex and low-resource language.

In a field like machine translation where state-of-the-art models are rapidly moving towards a direction with increasing demand on data and resources, we carry the hope that our dissertation can provide better insight on how to design more generic and efficient solutions that would allow to study many under-represented languages and be deployed by institutions who do not have access to expensive computing resources. All of the related software implementing our models are available for public usage.

Bibliography

[1] Muhammad Abdul-Mageed, Mona Diab, and Sandra Kübler. Samar: Subjectivity and sentiment analysis for Arabic social media. *Computer Speech & Language*, 28(1):20–37, 2014.

[2] Mohamed Afify, Ruhi Sarikaya, Hong-Kwang Jeff Kuo, Laurent Besacier, and Yuqing Gao. On the use of morphological analysis for dialectal Arabic speech recognition. In *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*. Citeseer, 2006.

[3] Kamla Al-Mannai, Hassan Sajjad, Alaa Khader, Fahad Al Obaidli, Preslav Nakov, and Stephan Vogel. Unsupervised word segmentation improves dialectal arabic to english machine translation. In *Proceedings of the Workshop on Arabic Natural Language Processing (ANLP)*, pages 207–216, 2014.

[4] Margaret Reece Allen. *Morphological investigations*. PhD thesis, University of Connecticut, 1978.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[6] Antonio Valerio Miceli Barone, Jindřich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. Deep architectures for neural machine

translation. In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 99–107, 2017.

[7] Joost Bastings, Wilker Aziz, and Ivan Titov. Interpretable neural predictions with differentiable binary variables. *arXiv preprint arXiv:1905.08160*, 2019.

[8] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

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

[10] Arianna Bisazza and Marcello Federico. Morphological pre-processing for Turkish to English statistical machine translation. In *Proceedings of International Workshop of Spoken Language Translation (IWSLT)*, pages 129–133, 2009.

[11] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, et al. Findings of the 2017 Conference on Machine Translation. In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 169–214, 2017.

[12] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of Computational Statistics (COMPSTAT)*, pages 177–186. Springer, 2010.

[13] Joan L Bybee. *Morphology: A study of the relation between meaning and form*, volume 9. John Benjamins Publishing, 1985.

[14] Mauro Cettolo. Wit3: Web inventory of transcribed and translated talks. In *Conference of European Association for Machine Translation (EAMT)*, pages 261–268, 2012.

[15] Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305, 2018.

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

[17] Noam Chomsky and Morris Halle. The sound pattern of english. 1968.

[18] Cho Kyunghyun Chung, Junyoung and Yoshua Bengio. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, 2016.

[19] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NeurIPS 2014 Workshop on Deep Learning*, 2014.

[20] Jonathan H Clark, Chris Dyer, Alon Lavie, and Noah A Smith. Better hypothesis testing for statistical machine translation: Controlling

for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: (Volume 2: Short Papers)*, pages 176–181, 2011.

[21] George N Clements and Kevin C Ford. Kikuyu tone shift and its synchronic consequences. *Linguistic inquiry*, pages 179–210, 1979.

[22] Ann Clifton and Anoop Sarkar. Combining morpheme-based machine translation with post-processing morpheme prediction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, Human Language Technologies (Volume 1: Long Papers)*, pages 32–42, 2011.

[23] Marta R Costa-jussà and José AR Fonollosa. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, 2016.

[24] Ryan Cotterell, Sebastian J. Mielke, Jason Eisner, and Brian Roark. Are all languages equally hard to language-model? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 536–541, 2018.

[25] Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. Labeled morphological segmentation with semi-Markov models. *CoNLL 2015*, page 164, 2015.

[26] Mathias Creutz and Krista Lagus. Unsupervised discovery of morphemes. In *Proceedings of the ACL 2002 workshop on Morphological and Phonological Learning*, volume 6, pages 21–30, 2002.

[27] Mathias Creutz and Krista Lagus. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR)*, volume 1, pages 51–59, 2005.

[28] Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):3, 2007.

[29] Hoang Cuong and Khalil Simaan. Latent domain translation models in mix-of-domains haystack. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, pages 1928–1939, 2014.

[30] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, Signals and Systems*, 2(4):303–314, 1989.

[31] Carl De Marcken. The unsupervised acquisition of a lexicon from continuous speech. *arXiv preprint cmp-lg/9512002*, 1995.

[32] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[33] Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 634–643, 2016.

[34] Philip Gage. A New Algorithm for Data Compression. In *The C Users Journal*, volume 12, pages 23–38, 1994.

[35] Paul L Garvin. The automation of discovery procedure in linguistics. *Language*, pages 172–178, 1967.

[36] Felix Golcher. Statistical text segmentation with partial structure analysis. *Proceedings of the Conference on Natural Language Processing (KONVENS)*, pages 44–51, 2006.

[37] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

[38] Andrew R Golding and Henry S Thompson. A morphology component for language programs. *Linguistics*, 23(2):263–284, 1985.

[39] John Goldsmith. An algorithm for the unsupervised learning of morphology. *Natural Language Engineering*, 12(04):353–371, 2006.

[40] John A Goldsmith. *Autosegmental and metrical phonology*, volume 1. Basil Blackwell, 1990.

[41] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[42] Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. Morfessor flatcat: An hmm-based method for unsupervised and semi-supervised learning of morphology. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1177–1185, 2014.

[43] Nizar Y Habash. Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187, 2010.

[44] Margaret A Hafer and Stephen F Weiss. Word segmentation by letter successor varieties. *Information storage and retrieval*, 10(11):371–385, 1974.

[45] Zellig S Harris. From phoneme to morpheme. *Language*, 31(2):190–222, 1955.

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

[47] Charles F Hockett. Two models of grammatical description. *Word*, 10(2-3):210–234, 1954.

[48] Petr Homola. Syntactic analysis in machine translation. 2009.

[49] Matthias Huck, Simon Riess, and Alexander Fraser. Target-Side Word Segmentation Strategies for Neural Machine Translation. In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 56–67, 2017.

[50] MichaelI. Jordan, Zoubin Ghahramani, TommiS. Jaakkola, and LawrenceK. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[51] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *ArXiv*, abs/1610.10099, 2016.

[52] Francis Katamba. *Morphology*. Macmillan International Higher Education, 1993.

[53] D Kinga and J Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, volume 5, 2015.

[54] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[55] Paul Kiparsky. Word-formation and the lexicon. Mid-America Linguistics Conference, 1982.

[56] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. Opennmt: Open-source toolkit for neural machine translation. *Proceedings of the 55th annual meeting of the Association for Computational Linguistics (ACL), System Demonstrations*, pages 67–72, 2017.

[57] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL, System Demonstrations*, pages 177–180, 2007.

[58] Kimmo Koskenniemi. Two-level model for morphological analysis. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, volume 83, pages 683–685, 1983.

[59] Ponnambalam Kumaraswamy. A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, 46(1-2):79–88, 1980.

[60] Mikko Kurimo, Sami Virpioja, Ville Turunen, and Krista Lagus. Morpho challenge competition 2005–2010: evaluations and results. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 87–95, 2010.

[61] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and label-

ing sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, volume 1, pages 282–289, 2001.

[62] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378, 2017.

[63] Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*, 2015.

[64] Pierre Lison and Jörg Tiedemann. Opensubtitles2016: Extracting large parallel corpora from movie and TV subtitles. 2016.

[65] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through L_0 regularization. In *International Conference on Learning Representations (ICLR)*, 2018.

[66] Minh-Thang Luong and Christopher D. Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, 2016.

[67] Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, 2015.

[68] Peter Hugoe Matthews and PH Matthews. *Inflectional morphology: A theoretical study based on aspects of Latin verb conjugation*, volume 6. CUP Archive, 1972.

[69] John McCarthy. A prosodic theory of nonconcatenative morphology. *Linguistic inquiry*, 12(3):373–418, 1981.

[70] Karuvannur Puthanveettil Mohanan. *The theory of lexical phonology*. PhD thesis, Massachusetts Institute of Technology, 1982.

[71] Eric Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders. In *International Conference on Learning Representations (ICLR)*, 2017.

[72] Sylvain Neuvel and Sean A Fulop. Unsupervised learning of morphology without morphemes. In *Proceedings of the ACL 2002 workshop on Morphological and Phonological Learning*, volume 6, pages 31–40, 2002.

[73] Kemal Oflazer. Two-level description of Turkish morphology. *Literary and linguistic computing*, 9(2):137–148, 1994.

[74] Kemal Oflazer and Ilknur Durgar El-Kahlout. Exploring different representational units in English-to-Turkish statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation (WMT)*, pages 25–32, 2007.

[75] William O'Grady, Michael Dobrovolsky, and Francis Katamba. *Contemporary linguistics*. St. Martin's, 1997.

[76] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, 2002.

[77] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[78] David Pesetsky. *Russian morphology and lexical theory.* PhD thesis, Massachusetts Institute of Technology, 1979.

[79] Tommi A Pirinen. Omorfifree and open source morphological lexical database for finnish. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 313–315, 2015.

[80] Maja Popović. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation (WMT)*, pages 392–395, 2015.

[81] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, 2014. PMLR.

[82] Robert Henry Robins. In defence of wp. *Transactions of the Philological Society*, 58(1):116–144, 1959.

[83] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[84] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

[85] Gozde Gul Sahin and Mark Steedman. Character-level models versus morphology in semantic role labeling. In *Proceedings of the 56th An-*

nual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 386–396, 2018.

[86] Hassan Sajjad, Francisco Guzmán, Preslav Nakov, Ahmed Abdelali, Kenton Murray, Fahad Al Obaidli, and Stephan Vogel. Qcri at iwslt 2013: Experiments in arabic-english and english-arabic spoken language translation. *Proceedings of International Workshop of Spoken Language Translation (IWSLT)*, 2013.

[87] Vıctor M Sánchez-Cartagena and Antonio Toral. Abu-matran at wmt 2016 translation task: Deep learning, morphological segmentation and tuning on character sequences. In *Proceedings of the First Conference on Machine Translation (WMT)*, 2016.

[88] Sunita Sarawagi and William W Cohen. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1185–1192, 2004.

[89] Patrick Schone and Daniel Jurafsky. Knowledge-free induction of inflectional morphologies. In *Proceedings of the Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–9, 2001.

[90] Philip Schulz, Wilker Aziz, and Trevor Cohn. A stochastic decoder for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1243–1252, 2018.

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

[92] Wolfgang Seeker and Özlem Çetinoğlu. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *Transactions of the Association for Computational Linguistics*, 3:359–373, 2015.

[93] Rico Sennrich, Orhan Firat, Kyungyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. Nematus: a toolkit for neural machine translation. In *15th Conference of the European Chapter of the Association for Computational Linguistics, System Demonstrations*, pages 65–68, 2017.

[94] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

[95] Dorothy Carla Siegel. *Topics in English morphology*. PhD thesis, Massachusetts Institute of Technology, 1974.

[96] Raivis Skadiņš, Jörg Tiedemann, Roberts Rozis, and Daiga Deksne. Billions of parallel words for free: Building and using the eu bookshop corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, pages 1850–1855, 2014.

[97] Noah A Smith. Linguistic structure prediction. *Synthesis lectures on human language technologies*, 4(2):1–274, 2011.

[98] Andrew Spencer. *Morphological theory: An introduction to word structure in generative grammar*, volume 2. Basil Blackwell Oxford, 1991.

[99] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[100] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3104–3112, 2014.

[101] Aleš Tamchyna, Marion Weller-Di Marco, and Alexander Fraser. Modeling Target-Side Inflection in Neural Machine Translation. In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 32–42, 2017.

[102] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. In *arXiv preprint arXiv:1605.02688*, 2016.

[103] Pieter Theron and Ian Cloete. Automatic acquisition of two-level morphological rules. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 103–110, 1997.

[104] Jörg Tiedemann. News from opus-a collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in Natural Language Processing*, volume 5, pages 237–248, 2009.

[105] Jörg Tiedemann. Parallel data, tools and interfaces in opus. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, volume 2012, pages 2214–2218, 2012.

[106] Francis M Tyers and Murat Serdar Alperen. South-east european times: A parallel corpus of balkan languages. In *Proceedings of the LREC Workshop on Exploitation of Multilingual Resources and Tools for Central and South-Eastern European Languages*, pages 49–53, 2010.

[107] Clara Vania and Adam Lopez. From characters to words to in between: do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2016–2027, 2017.

[108] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[109] Linlin Wang, Zhu Cao, Yu Xia, and Gerard de Melo. Morphological segmentation with window lstm neural networks. In *AAAI*, pages 2842–2848, 2016.

[110] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[111] Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 521–530.

[112] Chunting Zhou and Graham Neubig. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. In *Proceedings of the 55th Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 310–320, 2017.