Karlsruhe Institute of Technology

Neural Network-based Multilingual Translation Models

Master's Thesis of

Duc Tam Nguyen

at the Department of Informatics
Interactive System Labs

Reviewer:	Prof. Alexander Waibel
Second reviewer:	Dr. Sebastian Stüker
Advisor:	Dr. Jan Niehues
Second advisor:	Ms.-Inform. Thanh-Le Ha

Juni 1. 2015 – November 30. 2015

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

November, 30.,2015

. . . . . .

(Duc Tam Nguyen)

Abstract

Statistical machine learning usually requires extensive corpora for training. In the case of Machine Translation (MT) these corpora consist of large amounts of parallel texts in the chosen source and target languages. Since acquiring corpora is usually time-consuming and expensive, especially in the case of exotic language pairs such as Vietnamese- Indonesian, a third pivot language is sometimes used for a pipelined translation. I.e. a source language will be translated into the pivot language first, with the result finally being translated into the actual target language. However, in a straightforward pipelining systems like this the second MT system will tend to reinforce errors made by the first, leading to an overall poor translation quality.

In this work we attempt to transfer the idea of pipelining to the model level and combine this approach with an interlingua representation of language, leading to a **multilingual neural network based discriminative word lexicon** (NNDWL). These models are trained on various language pairs and are capable of translating between languages even without having seen **any sample sentences from that specific pair**. Our proposed multilingual architecture was able to outperform a simple pipelined unilingual NNDWL. Compared to unilingual NNDWL, which used translated sentences from source to intermediate language as input, the multilingual counterpart resulted in similar translation quality. If only the intrinsic objective function is considered, the error rates were even nearly identical.

This approach allows for translation between rarely looked-at languages without having to train a dedicated MT system for the specific pair. Moreover, using such a simple model, the system is already capable of generating a translation fairly quickly. In future works, we would like to adapt the concept to more complex models, in order to improve the overall translation quality. In addition, the system could be extended via additional languages and input modals, forcing the model to learn abstract or interlingua representation from the input data .

Zusammenfassung

Statistisches maschinelles Lernen benötigt für gewöhnlich große Trainingskorpora. Im Falle der Maschinellen Übersetzung (Machine Translation, MT) bestehen diese Korpora aus zahlreichen Paralleltexten in der gewählten Quell- und Zielsprache. Da das erstellen solcher Korpora zeit- und kostenintensiv is, insbesondere im Fall exotischer Sprachpaare wie z.B. Vietnamesisch-Indonesisch, wird manchmal eine dritte, sogenannte Pivot-Sprache verwendet um eine sequentielle Übersetzungs-Pipeline zu schaffen. D.h. die Quell-Sprache wird zuerst in die Pivot-Sprache übersetzt, und das Ergebnis dieser ersten Übersetzung dann in die eigentliche Zielsprache. Allerdings neigt das zweite MT-System in solch einer Pipeline dazu, Fehler des ersten Systems zu verstärken, was zu einer insgesamt schwachen Übersetzungsleistung führt.

In dieser Arbeit versuchen wir, das Konzept des Pipelining auf die Modell-Ebene zu übertragen und diese Herangehensweise mit einer Interlingua-Repräsentation zu verbinden. Dies führt zu einem Modell welches als **multilinguales, neuronale Netz-basiertes Diskriminatives Wortlexikon** (NNDWL) bezeichnet wird. Diese Modelle werden auf verschiedenen Sprachpaaren trainiert und sind in der Lage, zwischen Sprachen zu übersetzen **ohne Trainingsbeispiele des spezifischen Sprachpaars** gesehen zu haben. Die vorgeschlagene multilinguale Netzwerkarchitekturen übertrafen ihr intuitiv kombiniertes Gegenstück (**pipelined NNDWL**). Im Vergleich zum unilingualen NNDWL, welches die übersetzten Texte aus dem ersten MT-system im Pipeline verwendet, wurde von multilingualen Modellen eine ähnliche Übersetzungsqualität erzielt. Gemessen an verwendeter intrinsischen Lernzielfunktion waren die Fehlerraten sogar nahezu identisch.

Diese Herangehensweise erlaubt es, zwischen sonst selten betrachteten Sprachen zu übersetzen, ohne ein MT-System für genau dieses Sprachpaar trainieren zu müssen. Überdies ist das System mit einem solch simplem Modell bereits in der Lage, relativ schnell eine Übersetzung zu generieren. In zukünftigen Arbeiten möchten wir das Konzept auf komplexere Modelle übertragen um die Übersetzungsleistung zu erhöhen. Zusätzlich, könnte das System mittels zusätzlicher Sprachen und Eingabemodalitäten erweitert werden, um es so zu zwingen eine abstrakte oder Interlingua-Repräsentation aus der Eingabe zu lernen.

Contents

List of Figures

List of Tables

1. Introduction

Contrary to the expectations that arose from the **Georgetown Experiment** in 1954, the machine translation problem had not been solved "within a couple of years". Research focus first shifted from direct to transfer approaches and finally to studying interlingua representations, such as bi-directional grammars from which sentences in a target language could be generated. However, due to the complexity of natural languages and the resulting difficulties in formalising them, in the late 1980s statistical machine translation (SMT) experiments carried out by the IBM Research labs soon outperformed previous approaches. This approach is still widely utilised in both commercial and research applications.

As is usually the case for machine learning problems, SMT requires a large amount of training data i.e. parallel texts in the chosen source and target languages. Hence, data sparseness is one of the most pervasive issues to be overcome in order to produce high quality translations. Unfortunately, this data sparseness is quite common with rare language pairs such as Indonesian-Vietnamese. One possible solution that does not require additional effort is the usage of a **pivot language**, for example English. Since there is usually a much larger quantity of aligned parallel text available for English than for other languages this allows for training two systems - one that translates from the source language to English, and one that translates from English to the actual target language.

Nevertheless, in a simple pipeline, the second Machine Translation system often reinforces errors made by the first, leading to an overall poor translation quality. Moreover, since the pipelining happens on the system level, the models in each MT-system only employ bilingual data. Therefore, the possible information gain from using multi-lingual data is not fully realised. So the question is, how could the models be adapted to properly utilise the data available in this scenario?

First, consider the research area of **Deep Learning**. "Deep" Neural Networks have been applied successfully to problems in Natural Language Processing, Speech or Object Recognition tasks etc. Although they are not well understood, they outperform many other state-of-the-art approaches in their respective fields. Recently, in [7] an effective translation model, the so-called Discriminative Word Lexicon was designed using a very simple, fully connected neural network architecture. Although discarding the syntactical information present in the data, it can still be considered a good starting point to observe the effects of adapting a model to a multilingual use-case.

1.1. Contribution

The approaches presented in this master thesis draw upon the work on Neural Network based Discriminative Word Lexicon (NNDWL) in [7]. On one hand, the NNDWL is eas-

ily accessible and implementable in legacy Machine Learning Framework due to its simplicity. On the other hand, it takes into account long-range context, which make it less correlated/related/comparable to other modern models for MT.

Based on this model, we propose two types of **Multilingual NNDWL**, which were both able to translate from various source to various target languages. They made use of the large amount of available bilingual parallel data by implementing the idea of pipelining translation on the model-level and outperformed the intuitive system-level pipeline architecture. For training, no sentence aligned corpora of the source and target languages were required. The model can be used as a stand-alone translation model, without the need of using more complex MT-systems in the background. Furthermore, an important advantage of this approach is the capability of a single model to translate from/to various languages.

If combined with more complex translation models, a light-weight multilingual MT-system can be created which might be capable of translating sentences from various source languages within a small time frame on very limited hardware. Furthermore, the proposed approach can be extended to learn from an arbitrary number of source and target languages, making the model more language-independent. Moreover, multimodal inputs could also be incorporated in similar ways to enable the learning models the extraction of abstract representation from information.

1.2. Structure of this work

In chapter 2 we will give a short introduction to modern SMT as well as Deep Learning. These are prerequisites for understanding core concepts of recent research on the topic, such as Discriminative Word Lexicon or its neural network based counterpart NNDWL, which are presented in chapter 3. Since the approaches provided in this thesis were based on multilingual NNDWLs, chapter 4 will explain the underlying idea in detail and compare the multilingual approach to those using monolingual models. Here we also explain why the changes made in the training procedure of the proposed architecture was necessary. In chapter 5 these models are evaluated by measuring internal learning objectives, as well as translation quality achieved by applying them to an actual MT-system. Furthermore, experiments utilizing various training methods will explain the evolution of the according model clearly. Finally, we summarize our work in chapter 6, providing a short discussion of the inferred knowledge and showing research opportunities for future works.

2. Background

2.1. Statistical Machine Translation

The statistical approach needs parallel corpus from source to target language for training. More precisely it requires at least parallel sentences, which will be automatically aligned on word (and often phrase) level. In contrast to the initially developed, rule-based approaches in Machine Translation, such a system assigns probability to a certain number of different hypothesis, given a word sequence in the source language. The most probable candidate will be outputted as the final translation of the MT-system. State-of-the-art systems are based on statistical approaches, which are built upon various features to generate and score hypothesis e.g. Language Model, Translation Model, Phrase Model, Reordering Model, Word Lexicon etc. Those are combined using a log-linear-model to produce a single score, which shall approximate the desired probabilities.

Advantages of the statistical approaches are their minimal requirement of manual effort and therefore its scalability , transferability to other language pairs, ease of combining different models by using a Log-Linear-Model. Moreover translation quality can be improved by increasing computation speed and amount of data. However, compared to ancient systems using predefined rules or transitional memory, much more parallel data (and usually computing power) is required.

2.1.1. Bayesian and Log Linear Model

In order to choose a most probable translation for a given sentence f^I, one can assign the likelihood of all translation hypothesis e^J due to **Bayesian equation** for conditional probability as follows:

$$p(e^J|f^I) = \frac{p(f^I|e^J) * p(e^J)}{p(f^I)} \tag{2.1}$$

$p(f^I)$ describes the probability of the given source sentence and is constant during the maximization process. Hence, the equation simplifies to:

$$e_{max} = argmax_e \frac{p(f^I|e^J) * p(e^J)}{p(f^I)} = argmax_e p(f^I|e^J) * p(e^J) \tag{2.2}$$

In order to determine the real underlining distribution structures, infinite amount of data is required, which is not true in practice. Thus, one can just use approximation of those structures, which always differ from the real models. Depending on the system design, data and the task, the model which produces $p(f^I|e^J)$ could be more correct and relevant

than the other model or vice visa. A simple approach to address this problem is assignment of artificial weights to those probabilities, which will be optimized in the end using development data. The equation therefore changes into a log-linear form:

$$e_{max} = argmax_e p(f^I|e^J)^{\lambda_{TM}} * p(e^J)^{\lambda_{LM}} = argmax_e \lambda_{TM} \log p(f^I|e^J) + \lambda_{LM} \log p(e^J) \quad (2.3)$$

As a generalization of the previous approach, a Log-Linear-Model can be extended by features, which address various, relevant aspects of an appropriate translation, such as word counts or orders etc. Generally speaking, the equation can be re-formulated to:

$$e_{max} = argmax_e \sum_{0}^{i} w_i h_i(f^I, e^J). \quad (2.4)$$

where $h_i(f^I, e^J)$ are scores produced by feature model **i**, based on the source sentence and target hypothesis and w_i are the weighting factor for those scores. In the next steps, several most important features will be explained.

Translation Model Given a word sequence in source language, it computes the probability of a hypothesis in target language. This can be calculated by using translational probability of word pairs (in a word table) or phrase pairs (phrase table) and additional assumption about independence among sentence fragments to combine them by multiplying. Such a model produces $p(f^I|e^J)$, which was mentioned before.

Language Model It controls the sentence's fluency. Concretely, it approximates the likelihood of a given word sequence in the target language. Considering the above Bayesian formula, it is responsible for producing $p(e^J)$. Usually, the probability of the hypothesis will also be taking into account as an additional feature for a modern MT-system.

Reordering Model Even correctly aligned phrases or words are not supposed to be at the same positions in source and target language, due to grammatical variation. The reordering model learns from training data how to arrange words in the target language properly. Afterwards, It can be used to generate hypothesis and score them due to their order's suitability.

2.1.2. Rescoring

Instead of incorporating new features into the generation process of the n-best list, new models can directly be used in the rescoring phase. Finding the weighting factors in the log-linear-combination of different feature models in 2.4 is called *rescoring*. In each iteration, the candidates from n-best list are re-ranked due to weights factors updates, which are derived from comparing current n-best-list ranking to target sentence corpus. By adding the new model to this process, more complex and computationally expensive models can be easily included. Usually, Minimum Error Rate Training is the traditional

and most popular algorithm for statistical machine translation [22]. However, newer improved methods, such as MIRA in [4] and [16] and PRO in [10], have been presented in the last years. One of them, the ListNet algorithm as introduced to MT in [21] is a promising technique; it was reported to improve the BLEU score up to 0.8 points and outperform MIRA, PRO and MERT by up to 0.3 points.

While MERT can be grouped into the point-wise and PRO into pairwise techniques, ListNet considers the entire list of feature models. It is applicable on more complex feature combinations than the log-linear variant and scales well even on a high number of models. In ListNet for MT, a derivable loss function was designed, that measures the difference between probability distribution function of referenced and hypothesized ranking by using cross-entropy. It can be computed as follows:

$$Loss(y^{(i)}, z^{(i)}) = -\sum_{j=1}^{n} P_{y^{(i)}}(j) \log(P_{z^{(i)}}(j)) \tag{2.5}$$

Whereby i stands for the $i-th$ candidate in the generated n-best list, y for reference and z for hypothesized score of corresponding candidate. Furthermore, n is the total number of different feature models, $P_{y^{(i)}}(j)$ is the reference probability of the j-th model to be the first ranked one. In analog way, $P_{z^{(i)}}(j)$ specifies this probability due to current hypothesis.

Since this equation is partially derivable, optimization can be done using Stochastic Gradient Descent.

2.1.3. Evaluation metrics BLEU

BLEU-scores are automatic metrics to measure the translation quality of a statistical system and one of first metrics showing high correlation to human-evaluated translation quality. It could be determined using following equation:

$$BLEU = \min\left(\exp^{1-\frac{r}{c}}, 1\right) \exp(\sum_{1}^{N} w_n \log p_n) \tag{2.6}$$

with r and c being the reference and candidate sentence's length, N maximal number of n-grams being considered, w_n weight for each n-grams-score p_n, which are computed using precision scores on candidates, based on the reference list. Applying the logarithm function to the equation upon, it leads to:

$$\log BLEU = \min\left(1 - \frac{r}{c}, 0\right) + \sum_{1}^{N} w_n \log p_n \tag{2.7}$$

For more detailed explanation, please refer to [24]

2.1.4. Pivot translation

There exist about 5000 different languages on earth. In order to provide direct translation from a language A to B, $(5000 * 5001/2)$ different pairs have to be considered. Unfortunately, it is extremely difficult to acquire parallel data for a seldom used language pair,

such as e.g. Indonesian - German. A affordable approach makes use of an interlingua, intermediate language, which is more widely spread, such as English. In that case, the process is called "pivot translation", because the text will be firstly translated to the pivot language, and afterwards into the given, desired target language. The advantage of this approach is the dramatic reduction of the language pairs, which only have to be considered.

However, keeping in mind, that the Machine Translation problem is not completely solved and therefore the state-of-the-art systems still produce significant errors, those will be reinforced by pipelining 2 different MT-Systems to a single one.

2.2. Neural Networks

2.2.1. History of Deep Learning

Deep Learning has become popular in the last few years. Although Neural Networks were continuously developed, their popularity in recent researches was thank to the work of Hinton and his group in 2006. They enabled deep belief networks by training layer-wise greedily in [8]. Along with increasing computing power and cheaply affordable memory spaces, the trend of Neural Networks moves towards being larger, deeper and useful in different research field of interests, e.g. Natural Language Processing, Image Processing and Object Recognition tasks. In 2012, a deep convolutional network architecture, provided by Alex Krizhevsky et al. [15], outperformed state-of-the-art image recognition systems on the popular Image-Net-Tasks, and reduced the error rate until this time from about 25 % to the half. Since then, this special architecture is often referred as "Alex-Net", which is widely adapted by many researchers from different fields, like Natural Language Processing and Machine Translation as well. However, despite of not understanding exactly how neural networks could perform extraordinarily, neural networks architecture are in many cases a good decision of choice [1], when it comes to classification or regression task.

2.2.2. Network components

Many hyper-parameters have to be chosen for a neural network. For example, how many layers and neurons seem appropriate, how values should be forwarded through the network, which activation functions and learning algorithm should be used etc. This work only concentrate on some important components and aspects of Neural Networks. For detailed explanation, please refer to [1] and [2].

2.2.2.1. Network layers and their hidden units

In general, the larger, deeper the network is, the better the networks performance on training data would be. However, increasing the learning ability of a network does not necessarily lead to better generalization on not-seen data. This problem is known as Overfitting, i.e. the network is so well adapted to the training data, that it only performs very poorly on test data. Moreover, a more complex model means additional training time.

In the other direction, under-fitting can also occur. If the network is too simple, it might not be able to model the real, hidden structure of the given problem. Choosing a appropriate architecture along with their hyper parameters is the most difficult part in using neural networks. It requires different strategies and many empiric experiments.

A good design decision might be the usage of bottleneck features, which is done by adding a small layer (i.e. with a low number of hidden units, relatively to the count of neurons in neighboring layers) to the middle of the network. This way, the network can be forced to learn more meaningful representation of input data, which could lead to better overall performance. [7]. In this work, we oriented to previous researches in related fields and attempt to keep the network architecture as simple as possible.

2.2.2.2. Activation function

Sigmoid The Sigmoid-function is defined as

$$sigmoid(x) = \frac{1}{1 + exp(-x)} \tag{2.8}$$

The output lies between 0 and 1. It is useful to avoid out of range problems, once used in combination with Cross-Entropy or NLL-Criterion. However, by using many Sigmoid-activation function in a deep network, the vanishing gradient problem can occur.[1]

Tanh It can be easily shown, that Tanh is a re-scaled version of Sigmoid.

$$tanh(x) = 2g(2x) - 1 \tag{2.9}$$

The relevant aspect lies in the derivative function of Tanh, which is due to delta rule 4 times higher than Sigmoid's derivation. Hence, by using Tanh, one might does not increase the vanishing gradient problem as Sigmoid does.

ReLU It is a partially linear unit, whose use does not reinforce the vanishing gradient problem as the 2 previous ones. It became popular through the work in [15], which lead to nearly 50 % error's reduction of the state-of-the-art systems in 2012 .

$$ReLU(x) = max(0, x) \tag{2.10}$$

2.2.2.3. Objective function

MSE Minimum squared error for 2 vectors X, Y is computed as follows:

$$MSE(X, Y) = ||XY||^2 = \sum_i (x_i - y_i)^2 \tag{2.11}$$

It is well-known to be slow once applied to learning algorithm such as stochastic gradient descent.

Binary Cross Entropy For classification tasks, where there are only binary values, BCE is usually more preferred. It can be described as:

$$BCE(X, Y) = \sum_i -(x_i * log(y_i) + (1 - x_i) * log(1 - y_i)) \tag{2.12}$$

For more classes, a generalization of BCE, namely Negative Log Likelihood (NLL) can be considered. Please refer to [1] for more detail.

2.2.2.4. Learning Algorithm

The most popular and widely used method is Stochastic Gradient Descent (SGD) in combination with Back-Propagation. The later is a generalization of the delta rule; which computes the error gradients for each component of the neural network, given an input vector and w.r.t to a target vector. In SGD, based on those results on a given input sequence, parameter updates can be performed by moving them towards the direction, which is expected to reduce the error rate the most. This is done by choosing the opposite direction of the error gradients. [1] Implicitly, SGD make use of those result from one single training sample to approximate the real gradient-vector of the network w.r.t the given training data set. This process will be executed over a certain number of iterations, until the error rate is satisfied or the maximum number of epochs is exceeded.

2.2.3. Theoretical aspects about learning

Theoretically, by increasing neural network's complexity, the approximated function can be more complicated. However there are two different aspects to be considered here:

- Traditional learning approaches such as SGD could fail on very deep fully-connected neural network architectures. Although it is possible, an normal optimization process without any special mechanism results in an dead end of the parameter spaces i.e. even the training errors stay high, w.r.t. less complex networks. To face that problem, one could restrict the parameters spaces by using regularization mechanism (such as convolutional architecture) or LSTM [9] in case of infinite depth.
- From the theoretical point of view, the capability of a machine learning algorithm can be expressed using the Vapnik-Chervonenkis-dimension. For example, a deep network with a high number of free parameters is able to approximate very complex functions and has therefore a large VC-Dimension. By using this metric, the error's upper bound on test data can be approximated. Unfortunately, the higher the VC-dimension of an algorithm is, the larger its errors upper bound is. [27]. Hence, on one hand, it is reasonable to perform restrictions on the network to reduce it's capability, on the other hand, avoid under-fitting caused by a too simple model.

3. Related works

3.1. Discriminative Word Lexicon

Traditional features in a phrase-based MT-systems only make use of the bilingual context, which is bounded by the phrase's length on source and target side. Hence, in [17] and later in 2013 in [20] improvement of translation quality could be achieved by using the whole source sentence to predict target words. They proposed linear Maximum-Entropy models, which translates a sentence to a target language, given its source sentence. Furthermore, by extending the model with syntactical feature and feeding back errors from MT-systems, the quality was improved by up to 0.8 BLEU points.

More precisely, given a sentence from a certain source language, the Discriminative Word Lexicon predicts probabilities for words to appear in the target sentence. For each word, a single Maximum-Entropy model was trained. For this purpose, the training sentences from input language were described as bags-of-words, indicating which words from the vocabulary actually appears in the sentence. More formally, let s be the input sentence, $s = s_1 s_2 \dots s_n$ that comprises n-words. The representation is described by $F(s) = \{f_w(s) | w \in \text{source vocabulary } V\}$ with:

$$f_w(s) = \begin{cases} 1 & \text{if } w \in s \\ 0 & \text{if } w \notin s \end{cases} \tag{3.1}$$

On the target side, labels are generated in similar way, using:

$$label_{t_j}(s,t) = \begin{cases} 1 & \text{if } t_j \in t \\ 0 & \text{if } t_j \notin t \end{cases} \tag{3.2}$$

whereby the target sentence is described as t, and t_j are words of the target language's vocabulary. After training one linear ME-model for each target words, $p(t_j|s)$- the their conditional probabilities given a certain source sentence, can be approximated. Assuming all target words being independent, a sentence's probability can be computed by multiplying individual values of it's words:

$$p(t|s) = \prod_{t_i \in t} p(t_i|s) \tag{3.3}$$

By applying this approach, multiple occurrence of words will not be taken into account. However, since this method requires additional book-keeping which words were already considered and which not, a more efficient approximation of the previous equation was provided, with J being the targets' sentence length:

$$p(t|s) = \prod_{i=1}^{J} p(t_i|s) \tag{3.4}$$

The proposed models addressed long-range-context, since the entire source sentence was taken into account. However, structural information was dismissed, the input representations contain no word positions or similar features. Hence, the authors introduced additionally bags-of-ngrams, which were used to encapsulate small context of sentence's meaning. This was implemented in similar manner to bags-of-words, as in 3.1 and 3.2 Additionally, MT-errors were fed back to generate negative examples, leading to overall improvement of up to 0.8 BLEU points.

3.2. Neural network based Discriminative Word Lexicon

Motivated by results from [20] and popularity of Deep Learning methods, **Ha** in [7] applied multilayer neural network in order to replace the Maximum-Entropy model in the original DWL. He studied the impact of different quite simple architectures, which were intended to achieve bottle-neck-features. Overall translation performance was improved by up to 0.5 compared to the traditional DWL. By his work, the potential of Neural Networks w.r.t. improvement of translation's performance was shown up, which can be further exploited by using different, more complicate word representations, better adapted architecture e.t.c.

Similar to the traditional NNDWL, bags-of-grams representation was used. The binary numbers were arranged into a vector $v(s) \in 0,1^{|V|}$ with V being the vocabulary, s the sentence to be mapped. For the target side, $v(t|s)$ can be derived in similar manner. However, since the vectors were extremely sparse due to $|V| >> |s|$, there were negative effects on computation-time as well as memory-usage. Hence, the vocabulary was restricted to a certain number of *most frequent* words, with an additional placeholder for all other unknown words.

The model itself is shown in Fig. 3.1. It is symmetrical and the H_2 layer was supposed to be the Bottle-Neck-Feature (BNF) layer, by owning significantly less neurons than the neighboring layers. Except for the output layer, all layers were fully connected to its subsequent layer. Activation in the $i-th$ layer can be calculated as follows:

$$O^{(i)} = \sigma_i \left(W^{i^T} \left(O^{i-1} \right) \right) \tag{3.5}$$

With W^i being the weight matrix, σ_i the activation function, that was applied element-wise on the vector.

From a theoretical point of view, more complex dependencies can be modeled using a neural network than linear Maximum-Entropy models. Furthermore, parameters were shared among different target words by training only one single model, instead of one model for each word. This could be one of the reasons for improvement as shown in this work. Moreover, because of it's simplicity, training and evaluation process can be done very efficient, require no significant increase computation time.

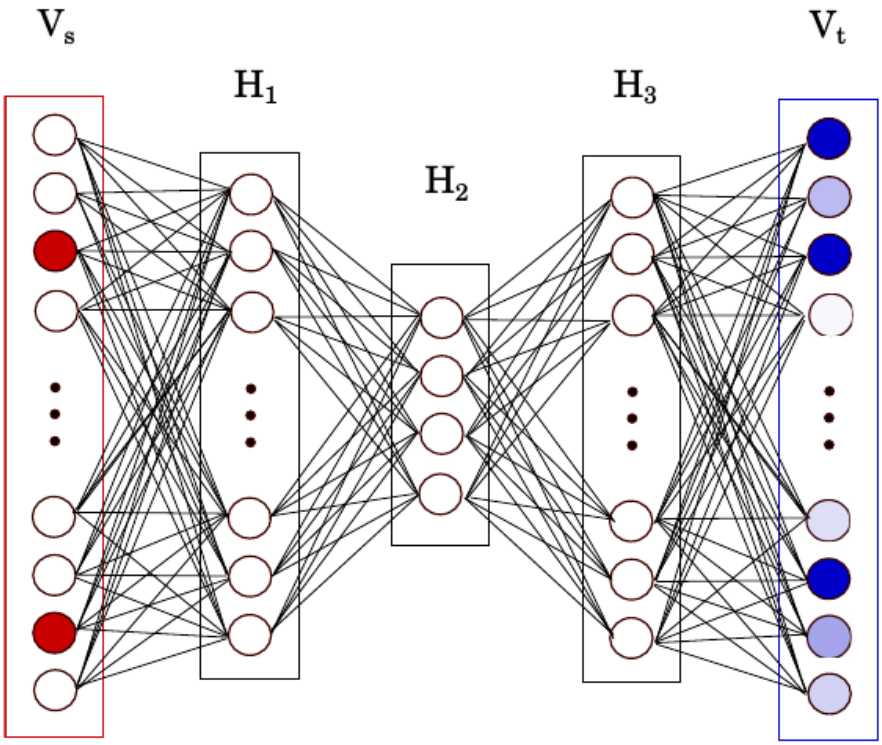


Figure 3.1.: Neural network based Discriminative Word Lexicon from [7]

3.3. Possible Extensions

3.3.1. In- and output representations

In order to apply a neural network to a typical Natural Language Processing problem like translation, the first step is to decide, how the raw texts can be mapped into a real-valued representation. Usually, this question is broken down to the representation of sentences, whose aggregation corresponds to the entire text. However, this decision depends on different aspects, such as the task, available hard- and software resources.

Bags-of-words The easiest but still very effective approach is using one-hot-vectors representation for words. By combining them, as used in [7], the resulting, constant-sized vectors only indicate if the words from corresponding vocabulary occur in the a sentence or not.

However, since the vocabulary may contain a huge amount of different words, it can cause memory and efficiency problems. This would slow-down experiments with different architectures, especially the hyper-optimization process. However, this approach is still preferred over some other representation like by using a binary vector. An important of such a one-hot vectored representation is the equidistance from each word pair in the vocabulary, which is not the case for binary vectors. A small distance between two unrelated word vectors could be interpreted as a strong connection between those two on

some level, which might lead to huge negative impact on the training process and lastly the overall performance of the network.

Vector of word indexes Instead of using any predefined word mapping, it is also possible to represent a sentence by a vectors of learn-able word embedding. The representation task can be transferred to a learning algorithm, for example a neural network. Let s be the sentence that has to be transformed, $v(s)$ the corresponding mapping, V the vocabulary,$v(s)_i$ the i-the element of vector $v(s)$. A sentence is then described as : $v(s) \in \{0,1\}^{|s|}, v(s)_i$ = position of $s[i]$ in V. I.e it is represented by it's words' indexes.

However, since the vector size is not fixed due to variable sentence length, traditional fully connected network is not applicable. In combination with mechanism such as convolutional network layer, this can be effectively be solved.

Additional features In additional to the bag-of-words features described before, it is also possible to represent a sentence by semantic preserving features like bags-of-n-grams or syntactical models e.g. Part-of-Speech-Tags etc. Nevertheless, to keep the essence of this work clear and simple, the main approaches are restricted to the use of bags-of-words-features.

3.3.2. Convolutional architecture

Convolutional Networks i.e. those, which owns at least one convolution layer, received increasing attention due to the fact, that they were the first, working deep architectures at all [1]. They have gained more attention after their work in [15], the proposed network reduced state-of-the-art error on the MNIST-dataset by the half.

Also known as Time-Delay-Neural-Networks (equal to convolutional network if the convolution operation is done through time), in 1989, this approach was developed in [28] for phoneme recognition. Firstly, instead of being dependent on the whole previous layer, each neuron's activation in the second layer is only influenced by input neurons within a small window. Secondly, for the next sequence of neurons, the applied weights are the same as before. In the other word, the previous was just slided to the next time unit along the signal. In total, this time-delaying-operations were performed at 2 hidden layers, which was followed by a summing-over-time operation.

Because of the special way of weight sharing, the proposed network resulted in transitional invariance and computational efficiency, since less free parameters remain to be trained. Thank to specified window size, a time context could be encapsulated which only seem to be relevant for production of a phoneme. Due to the pooling operation at last layer, it can even handle variable-sized inputs. However, due to the limits of computational power and available data at that time, the networks used were small compared to nowadays approaches.

Those ideas were adapted and applied to word tagging tasks from NLP by Collobert in [6]. The proposed network contains a layer for learning word-embedding, one convolutional layers and 2 fully connected layers as well. In detail, the input sentence was firstly transformed into a high dimensional spaces, spanned by it's word representations.

Afterwards, convolution filters were used, followed by a Max-pooling-layer. The output are then transformed using 2 linear combinations layers and their activation functions. However, despite its reported performance, convolution can be seen as a kind of regularization, which can be too strict and hence cause under-fitting due to [1].

4. Multilingual NNDWL

4.1. Idea

When providing an automatic translation system for a seldom language-pair by pipelining 2 systems via a pivot language, all applied features are restricted to the usage of the corpora from their corresponding language pair. In other words, the usage of multilingual data is happening on higher level of MT-Systems rather than of the models. Would it be desirable to incorporate multilingual information into the models directly to built up just a single MT-system ?

However, integrating multilingual information into all model used by state-of-the-art system is not a trivial task and perhaps not meaningful. In order to examine the effect, we started by investigating a simple, easily modified feature. The central idea of this work is to design a translation model, that profits from availability of parallel data in different languages. They can be used to enable the model to learn abstractions over them, leading the model to be language-independent.

In order to implement the idea, NNDWL was extended to fit the multilingual use-case. Traditionally, once the input vectors from a certain language are provided, the network produces activation in the hidden-layers, which are then combined to output corresponding word probabilities. In a general point of view, it is forced to learn to translate from a certain language to another. Now, imagine the situation, when it has to learn translations from high number of input languages to the original target language. Since the number of free parameters has not increased, the network has to come up with internal representations, which are valid for all input languages. When output language also varies, the networks' activation at hidden layers also have to be adjusted to abstract over them. In short, by sharing knowledge among different languages, the model's bias to a certain language might be reduced, hence making it language-independent.

This idea can be extended to make the models being not only language but also modal independent. For example, audio or image input can also act as inputs , since deep learning models have been reported to perform well on tasks from image and audio processing.

4.2. Contribution

4.2.1. Network architectures

4.2.1.1. Unseparated Joint model

The Neural network based Discriminative Word Lexicon model in Fig. 4.1 from [7] is limited to usage of parallel data from 2 language only. Given a sentence from the original

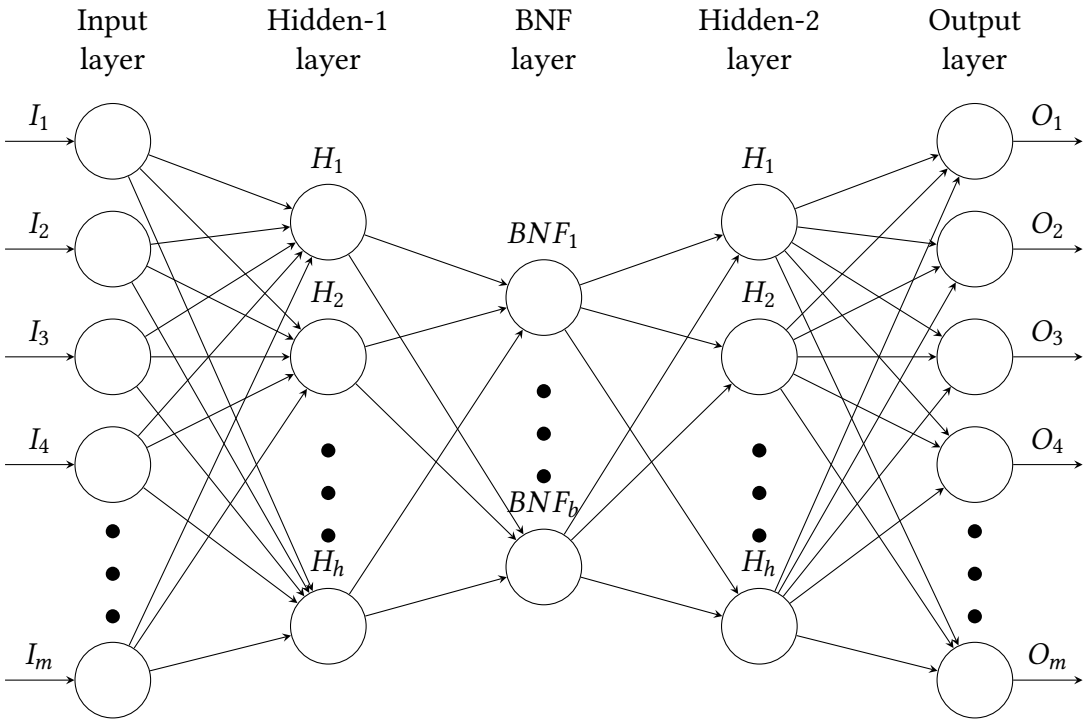


Figure 4.1.: NNDWL similar to [7]

side, it can be translated to a target language. If data from a different, third language is also available, it can not be fed into the network without any modifications, since the language's vocabulary cannot be the same. It's widely known, that no one-to-one, bijective word mapping exists between any language pair. Therefore, the place-holder neurons for a certain language, namely the input and output neurons in a traditional NNDWL, cannot be reused for another one.

This leads us to the architecture in fig 4.2, where in- and output neurons are divided into different sets, each responsible for a sentences from a certain language. This figure shows a model, which at least can process input from 3 different languages to produce also 3 languages. The number of in- and output neurons is now increased to $3*m$ whereby each m-sized set of neurons is reserved for it's corresponding language, indicating by I_{l_i}, $i \in \{1,2,3\}$. In this way, the network is able to handle data from different languages, without adding free parameters in the middle layers. Only at the in- an output layers, the number of weight parameters is now:

$$|weightsParameters_{input\&output}| = |languages| * m * h * 2 \tag{4.1}$$

m is the number of neurons assigned to each language and h size of hidden layers 1 and 2.

Network activation The next important question is, how the network can be used in our task. In such an architecture, inputs from 3 different languages are **always** required to produce any output vectors. The same is valid for the target side, once the error rate

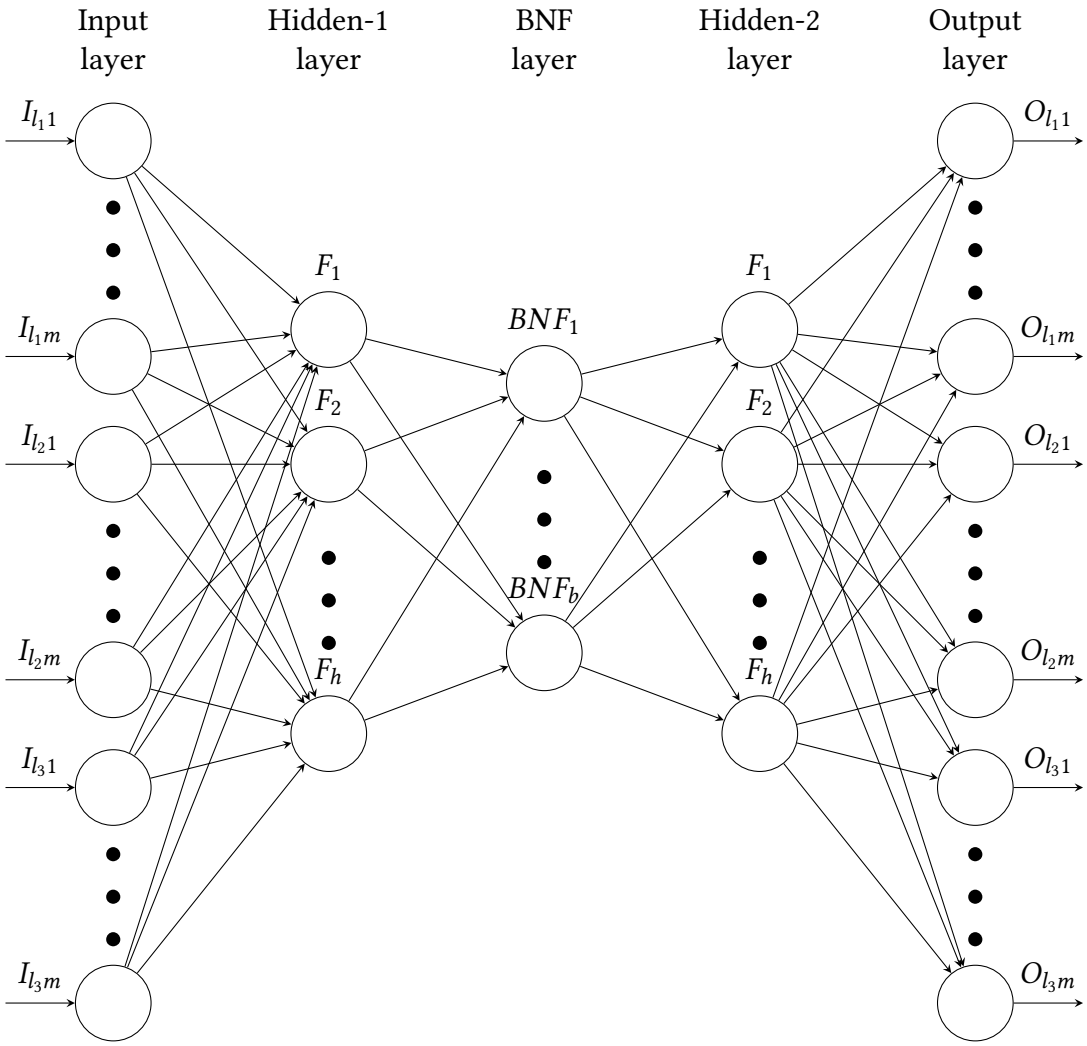


Figure 4.2.: Multilingual NNDWL

need to be computed. However, remind that in our task, the 2 available corpora for $l_1 - l_2$ and $l_2 - l_3$ are completely disjunct. In other words, there exist no input sentence, that is available in all 3 occurring languages. For example, if the network shall be trained from a sentence from $l_1 - l_2$, it is desirable to active the part of the network only, which are relevant. In this situation, activation of neurons from I_{l_2}, I_{l_3} as well as O_{l_1}, O_{l_3} have to be suppressed. This can be done by zero-padding the original input and target data. Furthermore, the error rate computed has to refer to the corresponding neuron set only (O_{l_2} in this example)

Pro and Cons The middle layers (more precisely the weights vectors between them) are shared among all languages, making the model not only multilingual but also in some way language-independent. By applying this simple extension, the model's bias to a certain language can be reduced. Moreover, in a translation pipeline for $l_1 - l_2 - l_3$ instead of taking some noisy translated sentences from a MT-system for $l_1 - l_2$, this model can translate from l_1 to l_3 directly. How this knowledge can be learned implicitly will be topic of 4.2.2.

On the other hand, the number of introduced parameters can not be neglected. Assuming $m = 5000$ and $h = 1000$, according to 4.1, the total amount of weights at input and output layers is: 15 millions for 3 languages instead of 5 millions. This would lead to much higher training time than usual. Another issue might be the language-dependent part, which is only reserved for each language. That is, at the input layer, relating to l_1 for example, only the linear combination part, spanned by 5 millions different weight factors in our example. The language-independent part might fail to abstract over languages, if the chosen mapping can not be compensated by previous layers.

In summary, the proposed architecture is 5-layers deep, the 3 hidden layers are supposed to be shared across languages and therefore language-independent, the input and output layers comprises different sets of neurons, whereby each set can only accept sentence from a certain language. Since the language dependent part only consist of a simple linear combination, it might be a too strict regularization, which causes under-fitting in the learning process w.r.t. the task. The name **unseparated** was chosen because of the forms of hidden layers in this approach.

4.2.1.2. Separated Joint model

The previously proposed architecture might have an possible weakness: the language dependent part of the network only includes a linear combination of neurons' values. However, since the correlations of sentence mappings of languages are highly non-linear, the models' modeling capacity can be too weak. Moreover, the language-independent part could become biased to one of the input languages.

This effect was reported in experiments with a neural network for multi-modal learning in [18]. By applying a similar architecture to audio and video data simultaneously, the independent layers tend to be strongly attracted by one of those two input modal. To face this problem, the authors extended the modeling capacity of the modal-dependent part by adding one additional, fully connected layer. In this way, the correlation among inputs mappings could better be modeled. Since the hidden layers are also divided into different parts, this model is identified as **Separated Joint Model**

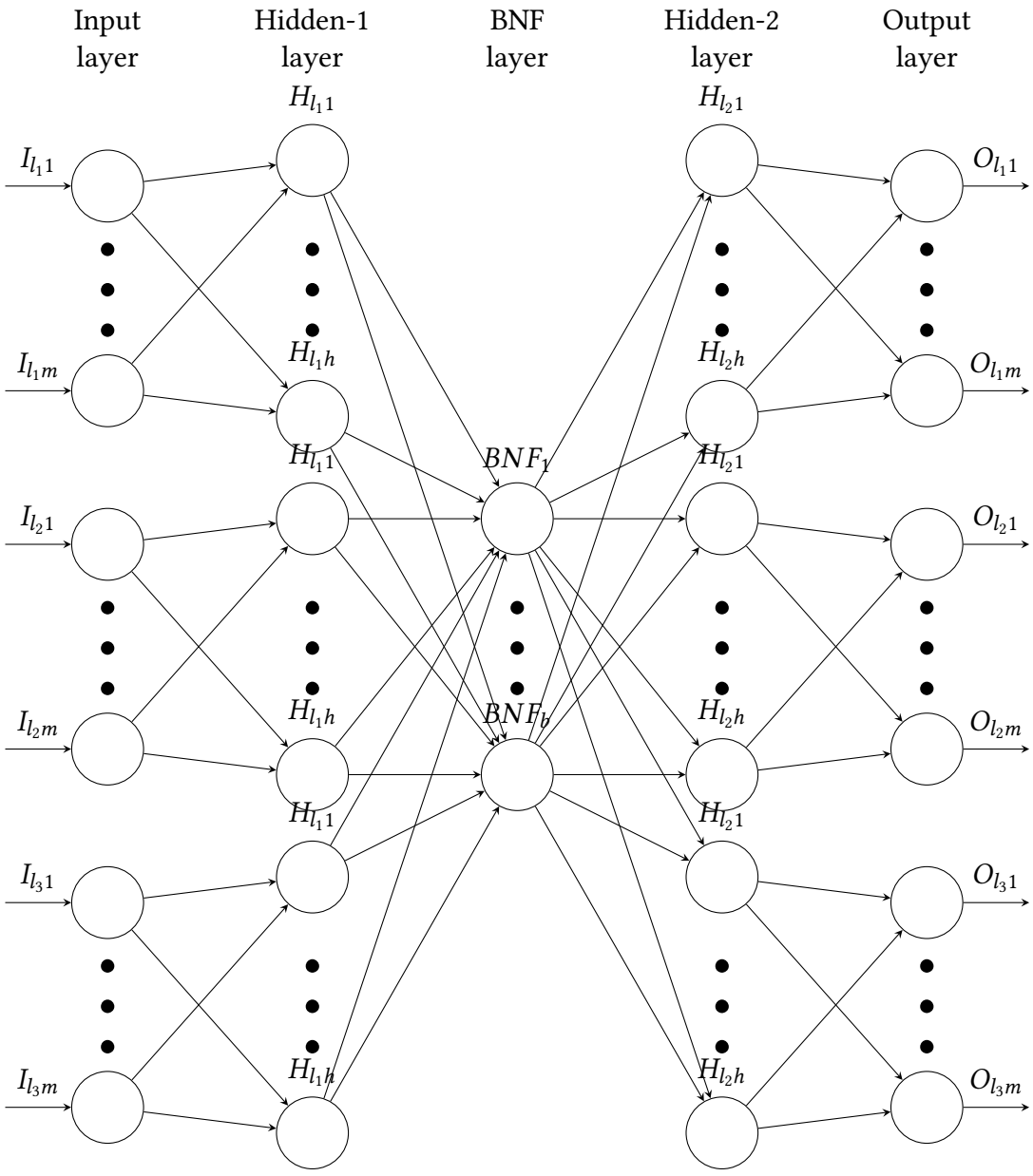


Figure 4.3.: Multilingual NNDWL with increased modeling capacities at in- and output layers

Although the data and the task differ from those in [18], theses ideas could be adopted to the current task. By increasing modeling capacity not only at input, but also at output layers, it lead to the network architecture in Figure 4.3. This way, the first and last two layers of the network are separated into different parts, which belongs to the corresponding languages. Due to this change, the only language independent part consist of the middle, bottle-neck layer, which is shared among all languages. So more free parameters remain left for each language's input. Notice that while the number of neurons at the 3. hidden layers stays constant compared to the first proposed joint model, the number of neurons at the first and third hidden layer is increased by 3. Intuitively, one may expect also these 2 numbers to stay constant, that is, for example on the input side, each language is then connected to a layer as wide as 1/3 of the first hidden layer in the not-separated joint model. Assuming each neuron can only produce binary output; then there already exist $2^{|Neurons|}$ different possible representation for input vectors. By reducing $|Neurons|$ to 1/3, the expression power of the first hidden layer will be dramatically reduced.

Pro and Cons Since the only, really shared parameters across all languages are only the bias at the middle layer, it is also reasonable to increase the capability of the language-independent part. A possible extension could be achieved by adding a hidden layer before and after the middle layer, to keep the network symmetrical (Encoding-Decoding-Architecture). However, a depth of 7 layers might lead to efficiency problem (owing to larger network) or may even not converge at all. In [15], deep network for object recognition which uses Sigmoid activation functions performed poorly. One of the reasons for that could be the *Vanishing/ Exploding Gradient Problem*, which also can occur in the proposed architecture, once it became too deep.

4.2.2. Training Procedure

Minimal data set In our task, parallel training data were only available for 2 language pairs: $l_1 - l_2$ and $l_2 - l_3$. One can start by using those data sets directly to train the proposed networks. However, this model will clearly fail to translate from l_1 to l_3, the 2 training sets appeared to be different tasks, from which the network has to learn. For example, assume l_1 stands for German (De), l_2 English (En), and l_3 French (Fr). Additionally, assume that a sentence such as "I like you" is available in all 3 languages. Nevertheless, by learning only from "Ich mag dich" -"I like you" and "I like you"- "Je t'aime", the network will even fail on translating "Ich mag dich" to French.

To understand why this is the case, w.l.o.g. let us consider the unseparated joint model as provided in Fig. 5.4.3.1. Since the sentence is represented in vector form and the word mappings from different languages are unrelated, there is no obvious connections between them. Hence, by inputting the German sentence, the network activation in the next layers differ strongly from those induced by the English sentence. Clearly, on the training set, the network will probably reduce the error rate close to 0 without having any problem. Nonetheless, once German input is applied, no useful French outputs can be expected, because the weights vectors connecting penultimate layers and French output neurons were not prepared to process this unknown activation. They were only trained

to handle those created by English input. In short, the model fails on translating from German to French.

The solution is to force the input layer to learn connections between different mappings and the output layer to process language-independent activation in the penultimate layer. This can be done by adding the same sentence from En-En to the training set. If the network is trained on En-En and De-En, the only different part is the connection weights at input layer. Ideally, the network should force 2 different mappings at input layers to produce the very similar activation in the first hidden layer and also in it's subsequent layers. This would make the middle layers be language-independent. Moreover, if the network is now trained on En-En and En-Fr, the connections at output neurons have to be able to handle the new representations which does not depends strongly on the input language.

Coming back to our task, remind that no sentence is available in all 3 languages. Only a certain set of words or small parts of sentences occurs in both data sets. Hence, the network has to learn from them and from a certain point of view, abstract over the meaning of training sentences.

In conclusion, the minimal training data set that is needed comprises : $l_1 - l_2$, $l_2 - l_2$ and $l_2 - l_3$ whereby $l_2 - l_2$ acts as a connector between the remaining data sets.

Learning generative model simultaneously In addition to the minimal training set as supposed before,$l_1 - l_1$ and $l_3 - l_3$ can also be considered, since they also affect translation from $l_1 - l_3$ strongly. This idea results from another concept of neural networks: **pre-training**. By forcing the network to reproduce the inputs themselves (generic model), it also improve the network's performance on the discriminative task. In many applications, it was applied successfully. Hence, in this work, those ideas were also evaluated compared to the minimal data set.

5. Evaluation

5.1. Data

For the later evaluation, parallel corpora from 3 different languages namely : German, English, French were chosen. Notice, that they will be identified by their abbreviation such as: De (German), En (English) and Fr (French). By referring to a combination like dataset De-Fr, the parallel text corpus for the language pair from German to French was meant.

The entire training, development and test data for those language pairs *De-En*, *En-Fr*, *De-Fr* were taken from the *Web Inventory of Transcribed and Translated Talks*, provided by [3]. They were used for the International Workshop on *Spoken Language Translation*, in the category *Machine Translation* in 2014. In order to keep the experiments across languages being comparable, the data from given corpus were only used, if they occurred in **all** datasets of the 2 desired languages pairs De-En and En-Fr. For example, if there was a training sentence such as *I like you* in the English data set, it remains in this set if and only if there was also appropriate oracle translation like *Je t'aime* in the French and *Ich mag dich* in the German data set.

After applying this restriction, the test data only contains 539 different segments, not sufficient to derive statistically stable inferences from the results. For that reasons, by crawling from the web for additional 3-language-parallel data, the test data set was extended to 3547 sentences. Table in 5.1 shows the numbers of segments in the final datasets.

Datasets	#Sentences
Training set	165723
Development set	887
Test set	3547

Table 5.1.: Overview of corpus size

Furthermore, the training data for each languages were also split into disjunct parts, calling *HALF-1* and *HALF-2*. The *HALF-1* data set contains the first 82861 segments and the rests were included in the *HALF-2*. By using only *HALF-2* (De-En) and *HALF-2* (En-Fr), one can simulate the situation of having no parallel data from German to French, but only De-En and En-Fr.

Preprocessing data for Neural Network Models Before being used as data for the neural networks, all non-stop-words were filtered out. The set of the remaining words of the training in each language and their mapping to a unique integer, ranging from 1 to the

set's size, was referred as the *vocabulary* of the corresponding language. Furthermore, since a neural network model can only handle real-valued input and output, an appropriate choice had to be made.

One possible choice is the use of bags-of-words to ensure constant length of input and output vectors. In that case, each sentence is represented by a vocabulary-sized-vector, with some *1*-values indicating which words were appearing in the sentence. Due to poor performance of this approach as reported in [7] and it's memory-wasting property, the size of vocabulary used in this work was reduce to 5000 **most frequent** words plus 1 Unknown word *UNK*. Once, the neural net requires source context additionally, they will also be preprocessed in similar manner. That is, the corresponding data will be represented as bags-of-ngrams, whereby the vocabulary's size of all existing n-grams was also restricted. For example, if a network is identified as *NN-5000-2000-500*, it means that the vocabulary on the source side should contain for uni-grams 5001, bi-grams 2001 and tri-gram 501 most frequent elements.

More flexible architectures, such as convolutional neural networks can also handle dynamical input's length. For that reasons, on the source language's side, a sentence will be mapped directly into it's real-valued representation by taking it's word's mappings, provided by the corresponding vocabulary.

In order to keep all architectures being comparable, the representation of the data for target side also stays the same: bags-of-words, consisting of 5000 most frequent words.

5.2. Evaluation metrics

Binary Cross Entropy Error Rate Based on back-propagation of BCE errors w.r.t. target data, as described in 2.2.2.3, the neural networks will be trained. Since some correlation between this metrics and the outputs' impact on translation quality can be expected, development data will be used for hyper optimizing of different nets, also using BCE as the objective. It will be considered as an intrinsic evaluation metric on the given task.

BLEU-Score In the current setting, a new feature can either be integrated into the decoding process, i.e. taking part at generating the translation lattices and hence the n-best-list or only be applied on the generated list. The first is referred to as *Optimizing*/*Decoding*, the later *Rescoring*, which currently could be done by using various algorithms; such as Mert, KBMira, KitPro and ListNet [21]. Both methods result in candidate generations of input sentences in a certain source language, whose translation quality can be measured using BLEU-Score afterwards.

In this work, the proposed feature model, along with various models were mostly evaluated by integrating them into the rescoring process of existing MT-systems. Since the application of new feature models in the optimizing process was quite time-consuming, only few experiments were done using it.

5.3. Environment

5.3.1. Torch7

Inspired from translation quality's improvement by such a simple architecture as NNDWL, achieved in [7], it was advantageous to move them from Theano into a easier-to-use, matlab-like open-source framework, called Torch7. It is based on the language Lua, which owns a powerful C API, that makes the language being able to be used as an extension for C-applications. Moreover, it is also possible to control programs, which are written in C by using Lua. Since many popular, highly optimized numerical libraries are written in C, such as *BLAS*, *LAPACK*, Lua was an important and thoughtful choice of the authors [5].

Similar to Theano (which is based on Python), a simple wrapper exist to start with. For this purpose, *DP* is the corresponding part of *pylearn2* in Python. In addition to *dp*, the following torch package was used: *Optim*, which provides various, highly optimized implementations of learning algorithms such as SGD or second-order-methods)

Additionally, the lua-extensions package *Moses* and *Allen* were included. They provide very efficient implementations for functions as in the _ (*underscore*) library from Javascript. Furthermore, *lua-utf8-Starwing* was used to handle all possible printable characters.

5.3.2. MT-Systems

System's configurations The baseline MT-Systems used in this work were similar but slightly simplified compared to those in [26],which were submitted for the IWSLT from Karlsruhe Institute of Technology in 2014. The data were also preprocessed by pruning possibly mismatched or inadequate translation pairs, e.g. if the sentences' length on target and source side differ too strongly. Moreover, once German was the source language, compounds were split up into words [13]. Additionally, a smart case model was used for the first letter of each sentence. Afterwards, the preprocessed data were word-aligned by using GIZA++ Toolkit [23] and summarized into phrases using grow-diag-final-and heuristic [14]. The language model considers up to 4-grams and was smoothed by the Witten-Bell-method. For reordering issues, POS-based short [25] and long range [19] rules were used as well.

For the sake of comparison, Maximum-Entropy-based Discriminative Word Lexicon [20] and it's extension to using source context were considered. In the extended version, up to 3-grams were taken into account.

5.4. Experiments & Results

5.4.1. Unilingual NNDWLs

These following neural network architectures were evaluated on the En-Fr-Half-2 dataset. The ideas proposed in [7] were compared to more standard approaches such as the Collobert architecture for word-tagging tasks [6] and their deeper convolutional counterparts. A detailed translation quality analysis of the En-Fr-Half-2 MT-systems after ap-

plying different architectures can be found in Tab.5.2. For this purpose, 4 rescoring algorithms were considered: *ListNet*, *KitPro*, *KBMira* and *MERT*. In order to reduce the statistical instability induced by these methods, the results were averaged and put into direct comparison to the intrinsic evaluation using Binary Cross Entropy criterion in Table 5.2.

In the following sections, the network architectures will be described more precisely. At the end, results were resumed and evaluated.

	BCE		**BLEU**	
	Dev	**Test**	**Dev**	**Test**
Baseline	-	-	22.00	31.43
+NNDWL-5000	52.68	40.55	0.14	0.20
+NNDWL-5000.2000.500	54.13	41.24	0.09	0.25
+NNDWL-5000.5000.5000	56.24	43.15	0.17	0.23
+NNDWL-collobert	63.69	52.95	0.07	-0.05
+NNDWL.2ConvLayers	60.98	50.11	0.05	0.02

Table 5.2.: Overview of NNDWLs performance, after training on En-Fr-Half-2-Dataset, measured by BLEU-score (higher is better) after rescoring and Binary Cross Entropy criterion after training (lower is better)

5.4.1.1. Network architectures

NNDWL using most frequent words for uni-grams In **NNDWL-5000** the number of most frequent words that shall be considered was restricted to 5000. By adding an optional tag for all remaining, not covered words by the vocabulary, the total number of words sums up to 5001. By using bag-of-words representation for input sentences, the number of input neurons also has to be 5001. For the output side, the same methods were applied, led the network to possess the same number of output neurons.

Furthermore, similar to the neural network in [7], It was decided to include 3 hidden layers + 2 (1 for the inputs and outputs each). One layer itself comprises a linear combination, which was followed by a activation function. The choice of neuron's number in each layer underlines some heuristics. For example, the third layer was supposed to be the bottle-neck-layer, which should enforce a effective, task-dependent coding of the inputs. This could be done by making this layer significantly smaller than it's preceding and subsequent layer. By designing the network symmetrically, the connections from 1. to 3. layer could be seen as a *encoding* path, the later somewhat like a *decoding* path. Moreover, for the rest of the network , number of neurons were chosen in way , that was to be sufficient for this task i.e. avoiding under and over fitting on the training data. This decision of network's depth seems appropriate, because deeper network might suffer from learning problems such as Vanishing or Exploding Gradient, which lead to not converging learning process at all.

In the case of activation functions, for each layer, a nonlinear one can be more or less arbitrary selected, depending on the desired behaviors on the network. However, since the network ought to produce some probabilities, it was recommendable to apply the Sigmoid activation function to the output layer. Furthermore, for the training process, the

network was initialized using normalized weight randomization. Additionally, Stochastic Gradient Descent was the learning algorithm of choice. A batch size of randomly selected 10 examples were used for training. In order to calculate the error rate for back-propagation, BCE-criterion was used. For the updating process, a learning rate of 0.002 was chosen. The whole training data set was shuffled every iteration to avoid bias to certain examples. After the limit of 70 epochs, the training process stopped and provided the best model according to the development set.

In conclusion, the architecture can be described as in Fig. 4.1. Notice that the first 4 layers will be followed by the activation function : $Tanh()$ and the last, output layer by $Sigmoid()$.

NNDWL + Source Context The models **NNDWL-5000-2000-500** and **NNDWL-5000-5000-5000** only differ to **NNDWL-5000** in the way how many input information will be used. Every remaining hyper-parameters stays the same. The input side was extended by additionally considering bi- and tri-grams. The size of the vocabulary for relevant n-grams was given by the number sequence. For example, 5000-2000-500 indicates , that the most frequent 5000 uni-grams (or words), 2000 bi-grams, and 500 tri-grams only will be taken into account.

Convolutional NNDWL (collobert) The network identified as **NNDWL-Collobert** was equivalent to the one used in [6] for general word-tagging tasks. Since it can handle flexible-length input vectors due to special convolutional structure, no selection of most frequent words has to be made as previously. The entire vocabulary can be considered by representing each sentence as a list of it's words mapping to a real valued number (given by the vocabulary).

Hence, the number of input neurons was not restricted to a fixed constant. Assuming an input sentence possesses 5 different words; in that case, it's representation consists of 5 different real-valued numbers. In the second layer, each number was transformed into a 40-dimensional space (word embedding size was hence 40). Our vectors can be viewer in 2 dimensional space with a size of $5x40$. Afterwards, zero-padding was applied before the first row and after the last column of the sentence representation. The number of zero-valued rows depends on the convolution filter's size, which was fixed to 3 . It can be determined using this formula:

$$\#zerosRowsOnEachSide = \left\lfloor \frac{FilterSize}{2} \right\rfloor \tag{5.1}$$

The resulting 2D-Tensor has a size of $7x40$. After applying an temporal (1D) convolution along the first dimension (#filters: 50, filter size: 3) the results' size was 5x50. A max operation over the first dimension lead to an one-dimensional vector (1x50). Afterwards were fed to a chain of 2 traditional linear combination layers, the first followed by the $Tanh()$ activation function whose output size was 1000, the second followed by $Sigmoid()$-function with an output size of 5000.

All other hyper-parameters of the learning process stays the same as in **NNDWL-5000**.

Convolutional NNDWL (deep network) Motivated by the trend of neural networks community to develop wider and deeper architectures, the **NNDWL-collobert** will be extended to the **NNDWL-2ConvLayers**. The word embedding size was now changed to 64. However, the most relevant change was that the network now possessed a pipeline of 2 convolutional layers; both of them made use of 128 different, learn-able filters. The first convolutional layer was characterized by a filter length of 3, followed by Rectified Linear Unit (ReLU), a zero-padding-mechanism as provided in 5.1 and a Temporal-Max-Pooling-layer with a size of 2. The second layer was similar to the one used in **NNDWL-collobert**, only the filter-size was 2 and the activation function was changed to ReLU. The remaining layers were exactly designed as in **NNDWL-collobert**.

5.4.1.2. Results

In table 5.2, due to BCE-Scores on development and test data, the evaluated architectures can be divided into 2 groups: fully-connected architecture (BCE ~40 on test data) and convolutional networks (BCE 52.95 and 50.11). The averaged BLEU-scores to measure translation quality also shows up similar behavior.

Within the fully-connected group, considering source context additionally seem to lead to worse BCE-scores (41.24 for NNDWL-5000.2000.500 and 43.15 for NNDWL-5000.5000.5000), however after applying models which make use of source context, the translation quality slightly improved. This indicates, that firstly, although there was some correlation between the BCE-score and BLEU-scores, it can not be stated for sure that better BCE-scores always result in improved translation quality. Moreover, since NNDWL-5000.5000.5000 led to better translation than both the NNDWL-5000 and NNDWL-5000.2000.500, one might expect higher improvement due to increasing number of most n-grams being considered, given appropriate architecture and hyper-parameters for learning.

On the other hand, the convolutional architectures seem to perform poorly. In addition to the baseline system, the Collobert architecture even decreased the overall translation quality on the test set. Although the network with 2 convolutional layers was slightly better according to BCE and BLEU scores, they were both far away from the performance of NNDWL-5000. Since convolutional layers can be seen as regularization, it's use seem to be too strict. It would lead to the under-fitting effect, in which the neural network's capacity was not high enough to learn the real function's complexity. Especially the Max-pooling-layer, which dismissed a high amount of (possibly) relevant information to be able to handle flexible-sized-input was a too strong restriction. Since the DWL-task was more difficult than the word-tagging-task, the network might require different, more sophisticated mechanisms to filter and preserve the relevant information.

In conclusion, even without considering source context, such a simple architecture like the NNDWL-5000 model still performed very well (+0.2 BLEU-score on test set). It can also be trained efficiently in sense of CPU-time and memory-usage. Moreover, transforming it into the multilingual case is quite straightforward. For that reason, in next section, scores of MT-systems using NNDWL in the optimizing process will also be considered. More importantly, an overview of MT-systems' evaluation on different data set will be provided, which shows up the weaknesses of a translation pipeline.

5.4.2. Optimization using NNDWLs

5.4.2.1. Baseline systems

As explained in 5.1, there were 3 different training data sets: the full dataset, and it's 2 disjunct parts, identified as Half-1 and Half-2. Based on the training set and the language pair, the resulting MT-systems performs differently on the test set, as shown in 5.3. The 3 columns *De-Fr*, *De-En*, *En-Fr*, along with the training dataset identifier, indicates the translation quality of a MT-system, trained on the corresponding dataset and language pair. The entries in De-En-Fr showed performances of a pipelined MT-System, in this case, it consisted of one system for each De-En and En-Fr. While both systems were trained on the same data in different languages for De-En-Fr (Full-dataset), the last entry in the table shows BLEU-score of a pipelined system, whose parts were trained on totally different corpus. The De-En system had only seen the first half and the En-Fr one only the second half of the corpus. Moreover, notice, that translation quality measured by BLEU can only be compared if the target data is the same. Hence, a system designed for De-En cannot be trivially compared to the one for De-Fr.

	System description	De-Fr	De-En	En-Fr	De-En-Fr
Full-Dataset	Baseline	21.03	27.98	32.73	20.48
HALF1-Dataset	Baseline	19.05	25.96	-	-
HALF2-Dataset	Baseline	19.11	-	31.06	18.48

Table 5.3.: Overview of translation quality of Baseline-MT-Systems

As shown in table 5.3, by doubling the corpus size, the translation quality induced by the systems were improved by about 2 BLEU-points, independent of language pair. Regardless of considering the full or half-dataset , both systems for De-Fr and De-En-Fr produced far worse translations than the system for En-Fr (> -11.5 BLEU score). It can be explained by taking account that German is more difficult than English, due to it's morphological richness and long-range dependencies.

Once the systems were combined to build up a translation pipeline, the quality decreases by 0.55 Point in case of training on full data set and 0.63 on Half-Half corpus. This is the entire gap between a direct and pipelined MT-systems.

5.4.2.2. Adding DWL-Models

In table 5.4, impact of Discriminative Word Lexicon features on the translation quality was shown. While the Maximum-Entropy-based DWL (MEDWL) considers the entire set of occurring words, the Neural Network based counterparts restrict the vocabulary size to 5001 (5000 most frequent words and an unknown word). In case of De-En-Fr, there exist 2 different models: NNDWL (De->En->Fr) and NNDWL (En->Fr). The former adapts the pipelining idea into feature level: by direct concatenation of the NNDWL-models trained for De-En and En-Fr. For this purpose, the English vocabulary used in both cases had to be identical. The later feature, namely NNDWL (En->Fr) was taken from MT-En-Fr. Since it was trained on En-Fr corpus, it wasonly able to accept English sentences as inputs, while the pipelined NNDWL (De->En->Fr) solely accepts German inputs.

	System description	De-Fr	De-En	En-Fr	De-En-Fr
Full-Dataset	Baseline	21.03	27.98	32.73	20.48
	+MEDWL	21.07	28.32	32.44	20.45
	+NNDWL (De->En->Fr)	21.07	28.47	32.36	**20.19**
	+NNDWL (En->Fr)	-	-	-	**20.38**
HALF1-Dataset	Baseline	19.05	25.96	-	-
	+MEDWL	19.4	26.51	-	-
	+NNDWL (De->En->Fr)	19.3	26.22	-	-
HALF2-Dataset	Baseline	19.11	-	31.06	18.48
	+MEDWL	19.17	-	30.75	18.44
	+NNDWL (De->En->Fr)	19.16	-	30.81	**18.37**
	+NNDWL (En->Fr)	-	-	-	**18.65**

Table 5.4.: Overview of quality of MT-Systems on test data after adding DWL-models into the optimizing process

After applying DWL-models to De-Fr-systems, significant improvement (+0.35-MEDWL, 0.25 NNDWL (De->En->Fr)) was obseved, once they were trained on the first half from the corpus. The translation quality did not change very significantly when trained on the full corpus or the second half. A totally different effect can be observed on De-En-corpus. Both NNDWLs lead to high improvement of quality, which indicates that the DWL being an appropriate complement to remaining models of that corresponding systems. However, the same statements are not valid for systems trained on En-Fr corpus. The application of both DWL- models decreases the translation quality by up to 0.27 BLEU-points.

In the case of the translation pipeline, De-En-Fr, the pipelined NNDWLs were outperformed by NNDWL (En->Fr) by 0.19 (on full corpus) and 0.28 (on half-half data set). This can be explained by looking at the way, how data wasforwarded through the pipelined NNDWL (De->En->Fr). Given a sentence in German, the first part of the network (NNDWL trained on De-En) produced word probabilities for all English words. While the most of them should be assigned to a very small number near to 0, there will not be numbers which are exactly 0 or 1. Moreover, since DWL only considers lexical dependencies on word level, it would induce not negligible errors. The second part, NNDWL taken from En-Fr, which solely were trained only on vectors consisting of 0 and 1, now had to handle real-valued inputs between 0 and 1. Moreover, the error rate of the first model was then reinforced by the later one, leading to overall suffering of translation quality (0.29-Full-Dataset, 0.11 Half-Half). In contrast to that, the NNDWL (En->Fr) received well-defined sentences, that were generated by considering various aspects addressed by models from MT-De-En.

The following experiments with multilingual architectures were aimed to close the performance gap between NNDWL (De->En->Fr) and NNDWL (En->Fr) by incorporating multilingual data in a more efficient way than a simple concatenation. Since pipelined translations showed lower quality than outputs from direct systems De-Fr, the proposed

architectures might even help to avoid errors made by the MT-system De-En by translating from German input directly.

5.4.3. Multilingual NNDWLs

5.4.3.1. Evaluated systems

In the following experiments, the proposed multilingual architectures were evaluated against each other by adding them to a MT-baseline-system as provided in table 5.3. If not stated otherwise, the hyper-parameters used for training process were as specified in 5.4.1.1 The candidates to be considered were:

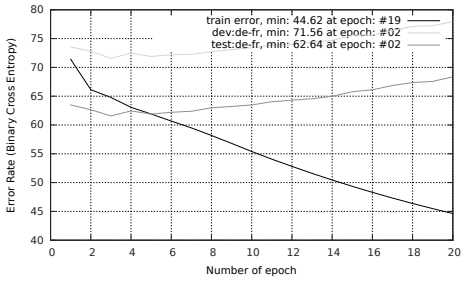
Baseline The basic MT-systems were configured as stated in 5.3.2. Depending on the experiments, the used train, development or test corpus might differ. However, it will be specified in the corresponding experiment and was valid for the succeeding analysis of different NN-DWL models.

NNDWL (En->Fr) The network's architecture was identical to those used in [7]. It uses English sentences as inputs and produces word probabilities for the French vocabulary. Once it was applied on pipelined MT-system (De-En-Fr), it made use of translated texts from De-En MT-system. This was the main drawback in contrast to multilingual models, since training a high quality MT-system is a difficult and time-consuming task.

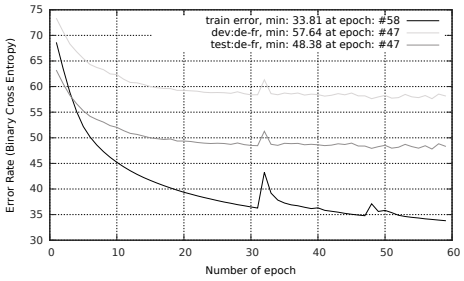
NNDWL (De->En->Fr) This model consist of 2 separately trained NNDWLs, each for De-En and En-Fr, which were afterwards simply concatenated. For that reason, this model uses German sentences as input.

U.JointModel The unseparated joint models share the same architecture as described in 4.2.1.1. Compared to the traditional NNDWL, the only difference was it's input and output layers, which consists of 3 sets of neurons, 5001 for each languages. Notice, that depending on the language of data in- and output, only the corresponding neuron set will be activated. Based on 2 corpus De-En and En-Fr, different combinations of data can be provided for the training phase of the network. *U. JointModel* indicates the minimal data set, consisting of De-En, En-En, En-Fr, which was needed to enable network to learn language connections between German and French. *U. JointModel+De-En* and *U. JointModel+De-En,En-Fr* etc. help to analyze the network's capability to learn language independent representations and it's impact on the overall translation quality.

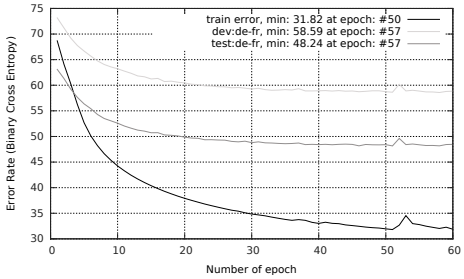
S.JointModel *U. JointModel*, *U. JointModel+De-En*, *U. JointModel+De-En,En-Fr* were designed and trained in the similar way as in 5.4.3.1. The only architectural differences show up in the second and the layer before the last layer. They were also separated into 3 different neuron sets, which were assigned to the corresponding language. Each neuron set comprises 1000 different neurons.



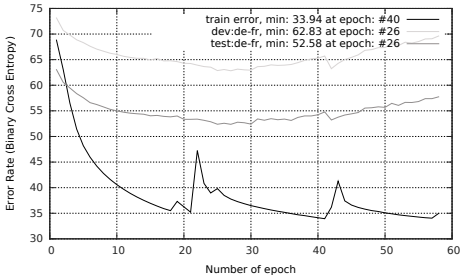
(a) Training of *U.JointModel* on De-En and En-Fr



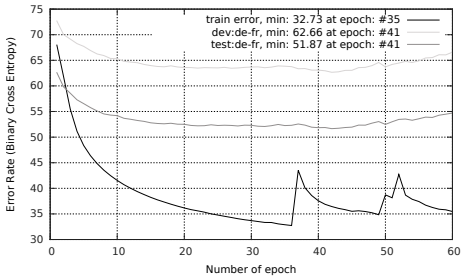
(b) Training of *U.JointModel* on De-En, En-En, En-Fr



(c) Training of *U.JointModel* on minimal dataset + De-De



(d) Training of *U.JointModel* on minimal dataset + De-De, Fr-Fr



(e) Training of *U.JointModel* on minimal dataset + De-De, Fr-Fr, Fr-En

Figure 5.1.: Multilingual NNDWL : Evolution on different training datasets

Random scores In addition to other evaluated models, also random scores were generated for the rescoring process. For this purpose, each candidate receives a integer, sampled from a pseudo-random, uniform distribution form $[0, 32767]$. Results induced by random scores was considered as a control group for the evaluations of models.

5.4.3.2. Different training procedure

In order to test effects of different training procedures on a multilingual NNDWL, the unseparated variant was considered. Note, that for the most models, the number of maximum was restricted to 60. Moreover, only 2 datasets were available, namely first half of De-En and last half of En-Fr.

Fig. 5.1a shows the situation where it learned from the data sets De-En and En-Fr. At the end of each epoch, translation quality for De-Fr was measured. Obviously, the network

failed to make use of training data; the error rates increased over the entire optimization process.

By providing En-En as a connector of both data sets, as shown in fig. 5.1b the network could reduce the test error rate to 48.38 BCE points during the training process. The network learned to abstract over meanings of words, since there was no parallel sentences in both datasets.

Following those ideas from pre-training as explained before, De-De was also added to the training procedure in fig. 5.1c or additionally even Fr-Fr in fig. 5.1d. While no significant change could be observed when added De-De, considering Fr-Fr appeared to have strong negative effect on translation quality. After the 30-th epoch, It was over-fitted to training data, loosing the generalization on dev. and test set as well.

Learning from Fr-Fr could be perceived by the network as a completely unrelated tasks analog to learning from De-En and En-Fr solely. Hence, fig. 5.1e shows smoother performance development on dev and test data if Fr-En was also taken into account. BCE-score was reduced by ~1 Point compared to training without En-Fr.

In conclusion, the minimal data set, consisting of De-En, En-En, En-Fr was confirmed to be the first working training procedure. Moreover, by adding different data set combinations, the network's behaviors according to BCE-metrics changed. In the next experiments, the network trained on De-En and En-Fr as well as its counterpart from De-En, En-En, En-Fr, De-De, Fr-En, Fr-Fr were no longer considered.

5.4.3.3. Unmatched corpus (Half-Half dataset)

In this situation, the pipelined MT-systems De-En-Fr consists of 2 systems, which were trained on completely disjunct data sets. MT-De-En takes the first half of data into consideration, while MT-En-Fr was relying on the latter half only. This should approximate the use case of having no parallel data, as usual in practice. In table 5.5 and 5.6, networks predictions were shown for 2 sample sentences. The words were ordered by their scores or in other words; occurring probability in the target sentence.

The first example sentence was "Everybody talks about happiness these days", which can be translated literally to "Aujourd'hui tout le monde parle du bonheur" (Aujourd'hui ~today, tout le monde ~the whole world, parle ~talks, du bonheur ~about happiness) In this example, the unseparated Joint models as well as NNDWL (En->Fr) performed very well and captured the most content-bearing words like parler (talk), bonheur (happiness), Aujourd'hui or maintenant (now, today).

The second sentence was "there is a lot of happiness coaching" which can be translated to "Il y a plein de techniques de coaching du bonheur" (Il y a ~there is, plein de ~a lot of) In this case, "techniques" and "coaching" were out-of-vocabulary words for all models. However, words such as beaucoup (a lot), bonheur (happiness), il y a were still recognized as most probable words. Moreover, all models could infer that at least one important, content-bearing unknown word was missing.

More detailed, overall comparison of performance induced by different models was evaluated in 2 different scenarios in the following. This resulted from the question, whether the components of a pivot translation systems were re-optimized or not after their con-

Source	Everybody talks about happiness these days.
Reference	Aujourd'hui tout le monde parle du bonheur.
NNDWL (De->En->Fr)	est le de UNK c la ce bonheur en du tout d
U. JointModel	UNK hui aujourd est de c monde bonheur parle ce parler d
U. JointModel + De-De	aujourd hui bonheur de d est UNK bien la ce les à
U. JointModel + De-De, Fr-Fr	aujourd hui bien la de à parler sujet qui le propos au
S. JointModel	est UNK c de le tout ce que les en d à
S. JointModel + De-De	est le UNK de ce jours aujourd hui tout bien les à
S. JointModel + De-De, Fr-Fr	est UNK c la qui de ce hui que à aujourd le
NNDWL (En->Fr)	bonheur le tout parle monde de est maintenant du c UNK

Table 5.5.: Example 1: Predicted words by different NNDWLs, ordered by assigned probabilities (descending order)

Source	There is a lot of happiness coaching.
Reference	Il y a plein de techniques de coaching du bonheur.
NNDWL (De->En->Fr)	beaucoup il y a très de UNK bien d en grand le les qu
U. JointModel	UNK a très beaucoup il y de est bonheur bien c en d une
U. JointModel + De-De	beaucoup très y il UNK de a bien la d souvent les là et
U. JointModel + De-De, Fr-Fr	beaucoup y UNK il très a de bien en d la peu avait tant
S. JointModel	beaucoup très UNK de a est il y en d bien les le un
S. JointModel + De-De	très beaucoup UNK il y de a une bien est le d un c
S. JointModel + De-De, Fr-Fr	beaucoup UNK très y il a de la d c est les le là
NNDWL (En->Fr)	bonheur il UNK y a de beaucoup le du d des un est la

Table 5.6.: Example 2: Predicted words by different NNDWLs, ordered by assigned probabilities (descending order)

catenation. In both cases, unilingual models were also considered together with multilingual counterparts, which were trained on different data sets.

Pivot translation The pipelining process can be done directly, without any systems knowing about each other. In this case, both of them were trained separately, relying only on data from their corresponding language pair. The development set used were also in clean format as provided in the corpus. Detailed rescoring results can be found in Tab. A.2. A shorter description was shown in Tab. 5.7.

	BCE		**BLEU**
	Dev	**Test**	**Test**
Baseline	-	-	18.68
+NNDWL (De->En->Fr)	64.21	53.06	0.11
+U. JointModel	57.40	47.78	0.12
+U. JointModel+De-De	**58.56**	**48.04**	0.07
+U. JointModel+De-De, Fr-Fr	62.83	52.37	**0.15**
+S. JointModel	**56.72**	**47.18**	0.10
+S. JointModel+De-De	58.15	48.23	0.02
+S. JointModel+De-De, Fr-Fr	59.98	49.32	**0.18**
+NNDWL (En->Fr)	**52.68**	**47.14**	**0.17**
+UniformNoise	-	-	-0.12

Table 5.7.: Translation quality of multi- and unilingual NNDWLs in addition to De-En-Fr-system (pivot translation)

When considering BCE-scores, NNDWL (En->Fr), U. and S. JointModel shows similar performance (Test score: approx. 47 BCE points). The second-best group was made up by training multilingual data on De-De additionally. Lastly, by including Fr-Fr, performance of unseparated JointModel decreases to the level of a pipelined NNDWL (De->En->Fr) (test score approx. 52.5). However, when the second layer was separated, overall performance only suffer slightly (test score 49.32).

After incorporating models' score into rescoring process, no significant improvement could be observed. In contrast to results induced from BCE-scores, by adding De-De and / or Fr-Fr, translation quality tended to be improved. Especially JointModel+De-De, Fr-Fr performed very similar to NNDWL (En->Fr) model. However, keep in mind that NNDWL (En->Fr) requires a complex MT-system to translate German sentences to English for further usage.

Re-optimized pivot translation The latter system MT-En-Fr in the translation pipeline can be trained on translated sentences, produced by the first system MT-De-En. However, due to lack of time, this work only considers a similar use case, that is, when the second system was hyper-optimized on translation output of the first system. Detailed rescoring results can be found in Tab. A.3. They were summarized by averaging by all algorithms, leading to an overview in 5.8.

	BCE		BLEU	
	Dev	**Test**	**Dev**	**Test**
Baseline	-	-	14.66	18.72
+NNDWL (De->En->Fr)	64.21	53.06	0.15	-0.01
+U. JointModel	57.40	47.78	0.20	-0.13
+U. JointModel+De-De	**58.56**	**48.04**	**0.22**	-0.08
+U. JointModel+De-De, Fr-Fr	62.83	52.37	**0.18**	-0.08
+S. JointModel	**56.72**	**47.18**	0.19	-0.13
+S. JointModel+De-De	58.15	48.23	0.12	-0.11
+S. JointModel+De-De, Fr-Fr	59.98	49.32	**0.17**	**0.04**
+NNDWL (En->Fr)	**58.10**	**47.14**	**0.14**	**0.11**
+UniformNoise	-	-	-0.25	-0.74

Table 5.8.: Translation quality of multi- and unilingual NNDWLs in addition to reoptimized De-En-Fr-system

The BCE-scores provided were identical to those in tab.5.8 and were supposed to help deriving dependencies between translation quality in this setting and intrinsic error rate measured by BCE.

The translation quality of the baseline was very similar to previous experiment, only changed from 18.68 to 18.72 points. On the dev. corpus, multilingual models outperformed the NNDWL (En->Fr) and NNDWL (De->En->Fr) by up to 0.08 BLEU points. Especially the dataset extended by De-De and Fr-Fr helped both models to perform stably on dev. set. Nevertheless, except for NNDWL (En->Fr) and S.JointModel + De-De,Fr-Fr, all remaining models resulted in decreased translation quality on **test set**. No model was able to outperform NNDWL (En->Fr) score, that is +0.11 BLEU score.

In conclusion, based on results from those 2 scenarios, some points could be observed:

- The proposed multilingual architecture were able to reach intrinsic performance of the unilingual NNDWL (En->Fr) , which itself required a complex MT-System for translation from German to French in background.
- In a normal pivot translation system, multilingual NNDWLs performed very similarly to unilingual NNDWL (En->Fr). However, if the latter systems MT-En-Fr was re-optimized on translated output from 1. system, multilingual architectures helped to improve translation quality at least on development dataset. However, the same behavior could not be observed on the test data set.
- There was no linear correlation between intrinsic and extrinsic evaluation metrics of translation quality measurement (BCE vs BLEU). Despite their small BCE error on test sets, the joint models trained on the minimal data set De-En, En-En, En-Fr were not always the best translation model.
- Although the separated model was less sensitive to addition of Fr-Fr data set, the unseparated models produced more appropriate scores for candidates from the n-best list

5.4.3.4. Matched corpus (Full-Full dataset)

In this experiment, both MT-systems which made up the pipelined translation system were trained on the same data from corresponding language pair. MT-De-En and MT-En-Fr both made use of the full, available parallel corpus. Based on this decision, 2 different scenarios were evaluated as previously: pivot translation with and without re-optimization.

Moreover, fewer models were taken into consideration due to lack of time and computational capacity. Uniform noise as well as U.JointModel and U.JointModel+De-De,Fr-Fr were omitted.

Pivot translation In this scenario, the latter MT-system, namely MT-En-Fr was not re-optimized on translated sentences of development set by MT-De-En. For comprehensive explanation, please refer to paragraph 5.4.3.3. Rescoring results using different algorithms were shown in tab. A.4 which were summarized and compared to internal metric BCE in tab 5.9.

	BCE		**BLEU**
	Dev	**Test**	**Test**
Baseline	-	-	20.77
+NNDWL (De->En->Fr)	60.67	49.91	**0.16**
+U. JointModel+De-De	56.36	46.56	0.10
+S. JointModel	**54.97**	**45.78**	0.13
+S. JointModel+ De-De	56.47	46.97	0.08
+S. JointModel+De-De, Fr-Fr	57.34	47.45	0.11
+NNDWL (En->Fr)	**49.53**	**44.24**	**0.16**

Table 5.9.: Translation quality of multi- and unilingual NNDWLs in addition to De-En-Fr-system

When considering BCE-scores solely, trends from previous experiment were confirmed. The simple concatenation of NNDWLs into one single model in order to implement translation pipeline on models' level led to worse performance (BCE 60.67 on dev and 49.91 on test data). The best model was again NNDWL (En->Fr) which relies on previous translation system. By training on De-En, En-En and En-Fr only, the network performs nearly as well as NNDWL (En->Fr). By training on additional data combinations such as De-De or Fr-Fr, networks' performance drops down by up to o 1.5 BCE points. Same behavior can be expected for not considered, unseparated models.

After applying models outputs to rescoring phase, no significant change could be observed by measuring translation quality of resulted sentences. All considered performed very similarly.

Re-optimized pivot translation Analog to experiment described in 5.4.3.3, the last part of translation pipeline (MT-En-Fr) was re-optimized on translated sentences of development set by MT-De-En. For detailed explanation, please refer to paragraph 5.4.3.3.

	BCE		BLEU	
	Dev	**Test**	**Dev**	**Test**
Baseline	-	-	16.82	20.72
+NNDWL (De->En->Fr)	60.67	49.91	0.06	0.12
+U. JointModel+De-De	56.36	46.56	0.12	0.11
+S. JointModel	54.97	45.78	0.08	0.16
+S. JointModel+ De-De	56.47	46.97	0.08	0.17
+S. JointModel+De-De, Fr-Fr	57.34	47.45	0.10	0.08
+NNDWL (En->Fr)	54.71	44.24	**0.10**	**0.12**

Table 5.10.: Translation quality of multi- and unilingual NNDWLs in addition to reoptimized De-En-Fr-systems

The BCE-values were the same as provided in tab 5.9 and were supposed to help improving the overview of this scenario. Compared to previous experiment, translation quality did not change despite of re-optimization on translated development set. On the test set, again, all multilingual NNDWLs achieved similar performance to NNDWL (En-Fr). No significant improvement could be observed, since S.JointModel and S.JointModel+De-De outperformed NNDWL (En->Fr) by only up to 0.05 BLEU points.

In conclusion, after these 2 experiments, some points were observed:

- Measured on BCE score, multilingual models perform as nearly as well as NNDWL which relies on previous MT-system, and much better than a simply pipelined NNDWL (De->En->Fr). However this effect could not be observed on translation quality measured by BLEU-scores.

- By re-optimizing on translated output from MT-De-En, no significant changes in translation quality were discovered

5.5. Further analysis

5.5.1. Multilingual models

This experiment was aimed at making results from section 5.4.3.3 more transparent. In this scenario, the applied multilingual models were trained on a unmatched corpus; namely Half-Half-corpus (first half of the data set was De-En, the other half En-Fr).
For an example sentence: "Everybody talks about happiness these days" considered in 5.11, different networks ranking did not vary much. The most models assign the best scores to the 6 identical candidates. Slightly different candidates were preferred by U. JointModel, S. JointModel and S. JointModel+ De-De, Fr-Fr, although they were not accurate as the other set of candidates.

Apparently, the multilingual models produce similar rankings for candidates from n-best list. Therefore, the number of considered samples were extended to **20** source sentences, score distributions were visualized for development data in fig. 5.2 and for test

Model	Highest scored candidates
NNDWL (En->Fr)	tout le monde parle bonheur .
NNDWL	tout le monde parle le bonheur .
U. JointModel + De-De	tout le monde parle le bonheur . "
U. JointModel + De-De, Fr-Fr	tout le monde parle bonheur . "
S. JointModel + De-De	tout le monde parle le bonheur " .
	tout le monde parle bonheur " .
U. JointModel	hui tout le monde parle bonheur .
	hui tout le monde parle le bonheur .
S. JointModel	tout le monde est parler bonheur .
	tout le monde est parler le bonheur .
S. JointModel + De-De, Fr-Fr	tout le monde parle heureux .

Table 5.11.: Overview of best candidates according to NNDWLs, given the source sentence "Everybody talks about happiness these days"

data in fig. 5.3. Given a certain source sentence, scores of the candidates were normalized to [0,1]. For NNDWL (En->Fr) the candidates were sorted according to their ranking, with the best candidate standing on the left side. All remaining models follow the ranking provided by NNDWL (En->Fr). In fact, these figures can be used to derive information about how **different assigned scores** of models were compared to NNDWL (En->Fr)

By looking at figure 5.2 and figure 5.3, obvious repeated patterns could be detected. Especially the ranking on the left side of the graphs were very similar to those induced by NNDWL (En->Fr). This means, that all NNDWL models agreed mostly about the question, which candidates were the best one. However, there were very strong variations in ranking of worse candidates. On the examples where NNDWL (En->Fr) could not distinguish clearly between good or bad candidates, other models also came up with very different rankings of candidates. This trend can also be observed in experiments with 10 or 50 examples (see Appendix A.2)

Moreover, a similar analysis was also applied to visualize network output probabilities for development set in fig. 5.4 and for test set in fig. 5.5. Again, NNDWL (En->Fr) act in some sense as a baseline, since word were ordered based ranking derived from occurring probability, produced by NNDWL (En->Fr). By looking at the left side of all graphs, one can see that the models agreed about a very small set of words, which might occur in the target sentence. The most probable words in the graphs were almost quite similar one, independent of which model was considered. Furthermore, it was observable, that NNDWL (De->En->Fr) was quite similar to NNDWL (En->Fr) while all other models produced strongly different word probabilities. Moreover, outputs from the Separated JointModels were brighter and had lower contrast, suggesting they were worse at distinguishing between more or less likely words.

Statistics about relative difference of different models and NNDWL (En->Fr) were summarized in 5.11. It was measured using Mean-Square-Error. As observed before, the outputs distribution of NNDWL (De->En->Fr) was significantly more similar to NNDWL (En->Fr) than other models. However, it's scores distribution were very far away from

NNDWL (En->Fr) (343.44 Points on dev., 280.14 on test data). Moreover, the more data set was considered, the farther the distance became, whereby S. JointModels were less sensitive to addition of training data. Furthermore, in general, outputs produced by unseparated model were more similar to those of NNDWL (En->Fr) than the remaining models.

In conclusion, by considering samples of 10, 20 and 50 sentences as well as overall ranking difference, some trends could be observed:

- The highest scored candidates were nearly always the same among all models. This can be resulted from the models' agreement about most probable words, given a certain source sentence.
- While candidate ranking provided by NNDWL (De->En->Fr) was very different from NNDWL (En->Fr), the JointModel trained on minimal dataset (De-En,En-En, En-Fr) was much more similar. Moreover, ranking induced by JointModel +De-De, Fr-Fr strongly differs from NNDWL (En->Fr), this could explain different effects on translation quality compared to NNDWL (En->Fr).

	Dev		Test	
	Word probability	Scores	Word probability	Scores
NNDWL (De->En->Fr)	1.61	**343.04**	1.42	**280.14**
U. JointModel	1.72	**167.70**	1.56	**133.19**
U. JointModel + De-De	1.72	180.35	1.52	150.66
U. JointModel + De-De, Fr-Fr	1.83	259.23	1.64	219.03
S. JointModel	1.65	**187.76**	1.49	**157.06**
S. JointModel + De-De	1.68	184.35	1.50	153.07
S. JointModel + De-De, Fr-Fr	1.67	256.96	1.51	201.45

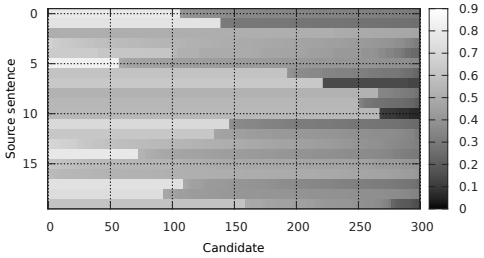
Table 5.12.: Relative distance of NNDWLs to NNDWL (En->Fr) measured by Mean-Squared-Error (MSE) per sample

5.5.2. Dev. and test dataset discrepancy

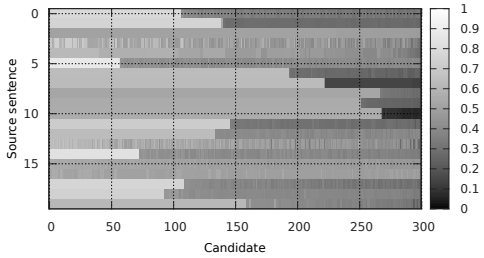
	Dev		**Test**	
	Mean	**Variance**	**Mean**	**Variance**
De	18.72	0.18	14.02	0.16
En	19.68	0.19	14.70	0.16
Fr	19.14	0.18	15.47	0.17

Table 5.13.: General statistical metrics applied on data sets

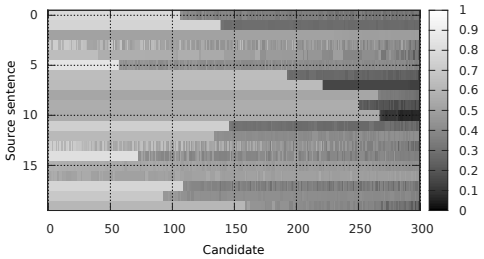
In the previous experiments, the networks' performance measured by BCE was on development data always much worse than on test data. In other words, the translation of



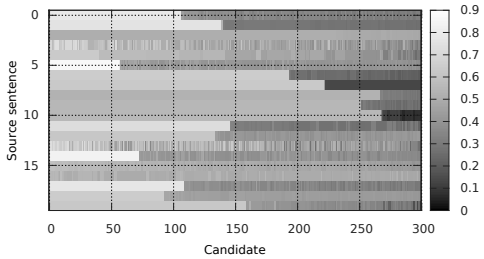
(a) NNDWL (En->Fr)



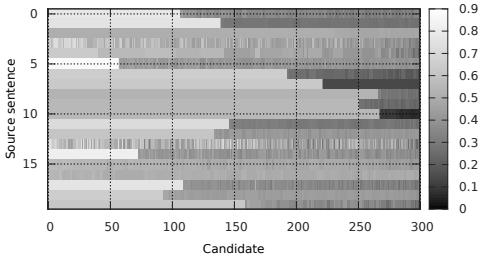
(b) NNDWL (De->En->Fr)



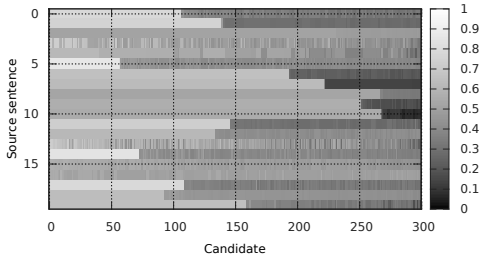
(c) U. JointModel



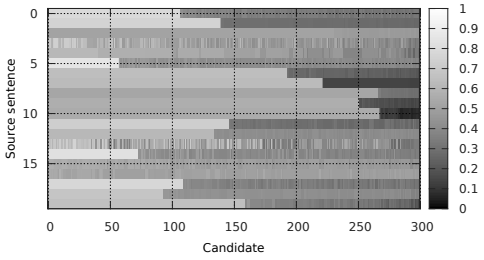
(d) U. JointModel + De-De



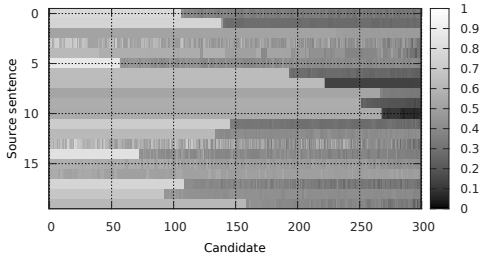
(e) U. JointModel + De-De, Fr-Fr



(f) S. JointModel

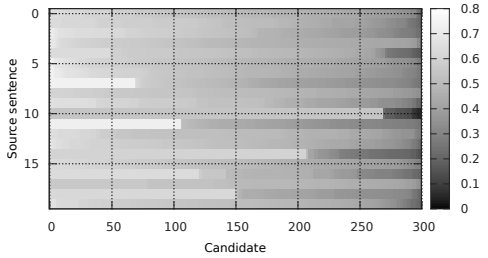


(g) S. JointModel + De-De

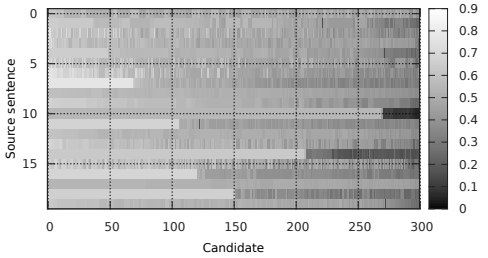


(h) S. JointModel + De-De, Fr-Fr

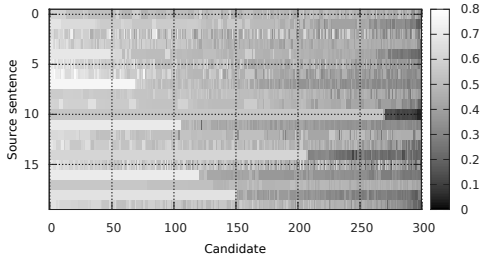
Figure 5.2.: NNDWL scored candidates for the first **20** sentences of development data set



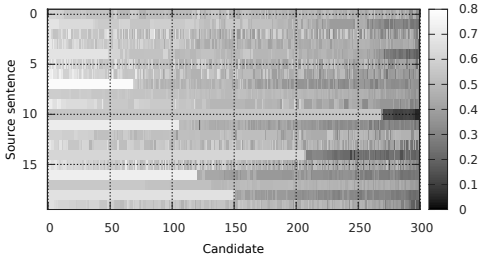
(a) NNDWL (En->Fr)



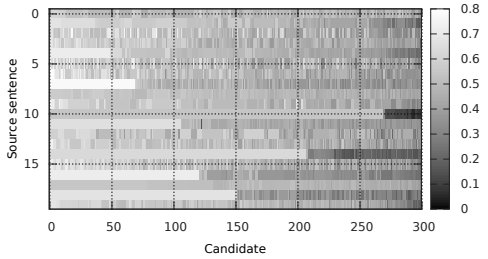
(b) NNDWL (De->En->Fr)



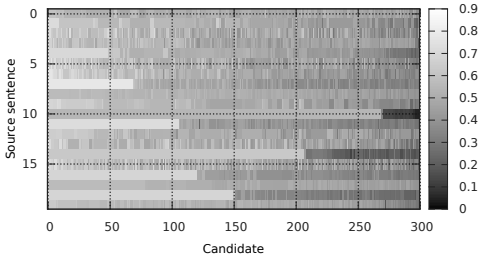
(c) U. JointModel



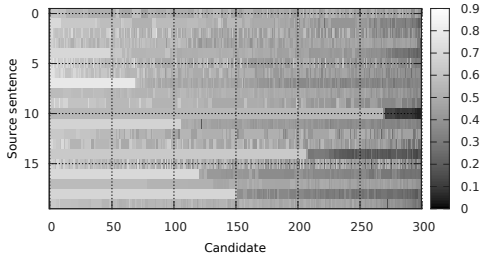
(d) U. JointModel + De-De



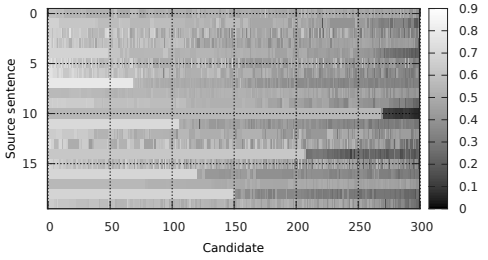
(e) U. JointModel + De-De, Fr-Fr



(f) S. JointModel

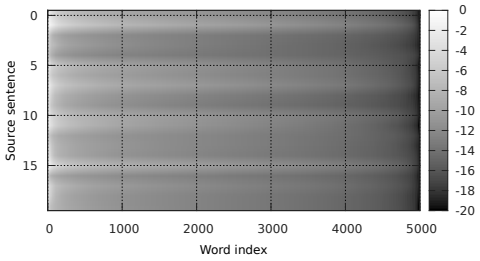


(g) S. JointModel + De-De

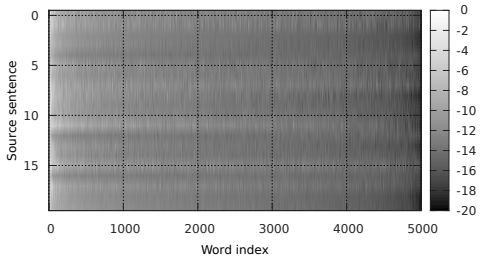


(h) S. JointModel + De-De, Fr-Fr

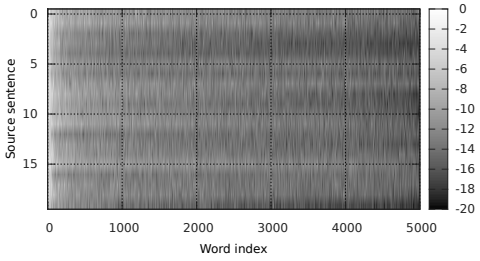
Figure 5.3.: NNDWL scored candidates for the first **20** sentences of test data set



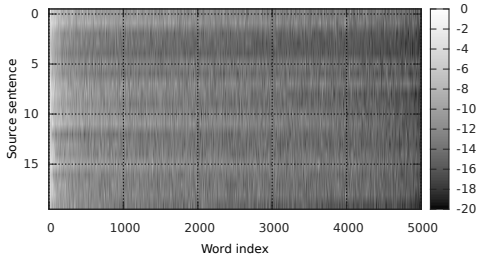
(a) NNDWL (En->Fr)



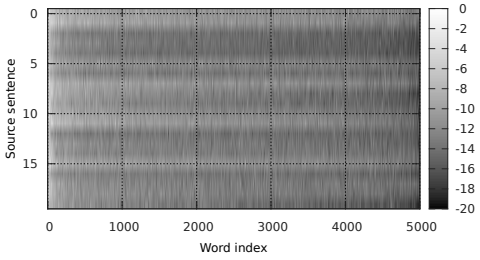
(b) NNDWL (De->En->Fr)



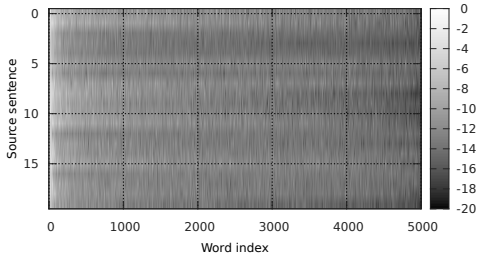
(c) U. JointModel



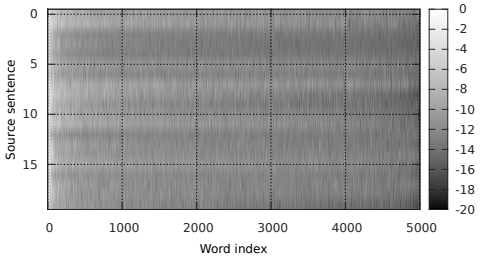
(d) U. JointModel + De-De



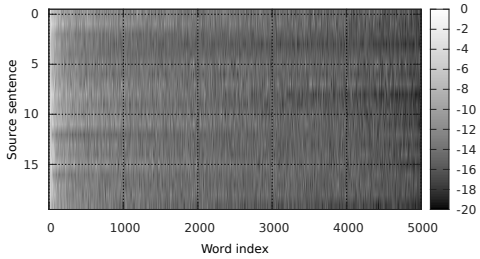
(e) U. JointModel + De-De, Fr-Fr



(f) S. JointModel

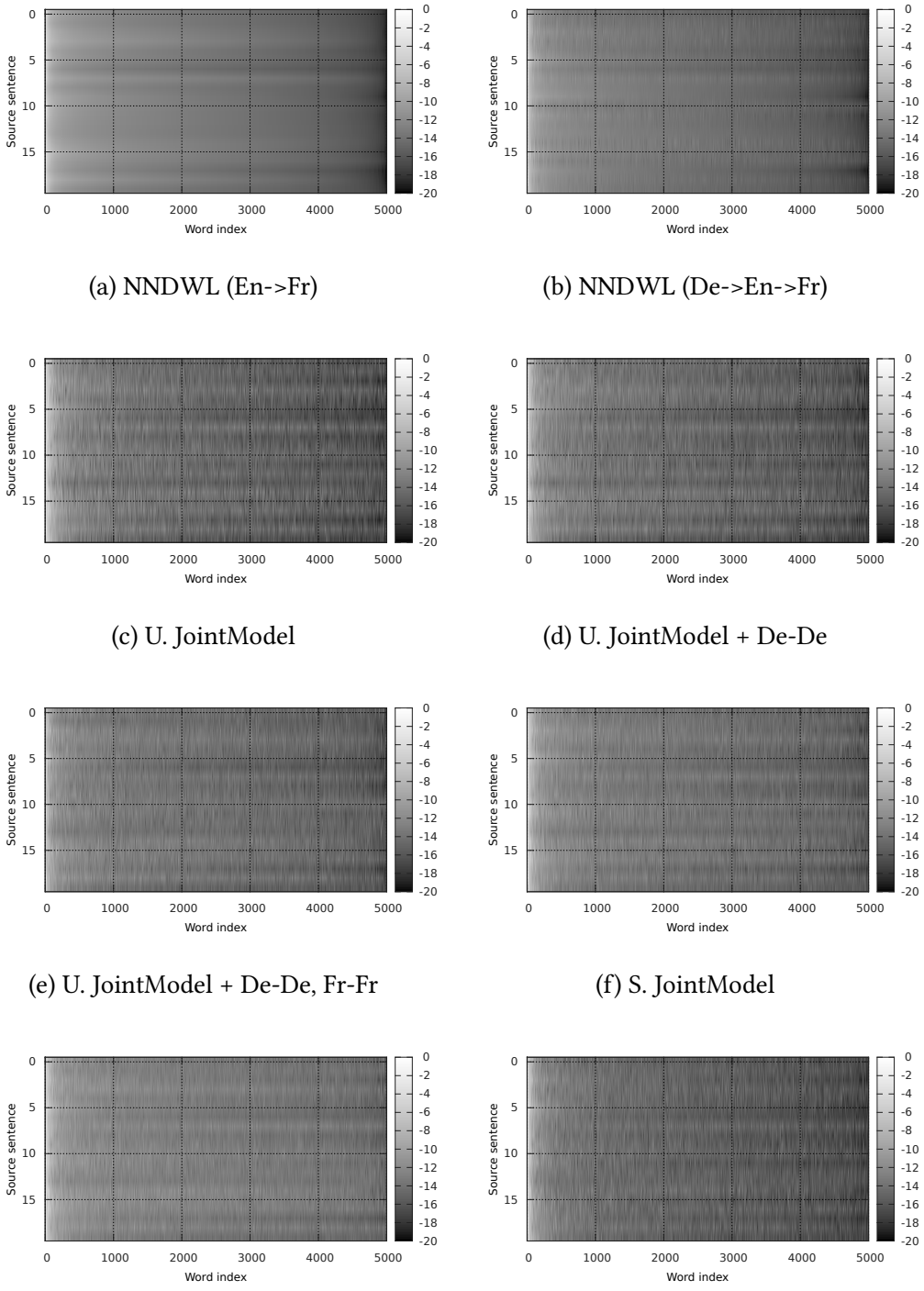


(g) S. JointModel + De-De



(h) S. JointModel + De-De, Fr-Fr

Figure 5.4.: NNDWL computed French words' probabilities , given the **20** sentences from development data set (log_2 scale)



(a) NNDWL (En->Fr)

(b) NNDWL (De->En->Fr)

(c) U. JointModel

(d) U. JointModel + De-De

(e) U. JointModel + De-De, Fr-Fr

(f) S. JointModel

(g) S. JointModel + De-De

(h) S. JointModel + De-De, Fr-Fr

Figure 5.5.: NNDWL computed French words' probabilities , given the **20** sentences from test data set (log_2 scale)

dev. data was more difficult than test data. Moreover, behaviors of multilingual models on translation quality using dev. data also differs from those on test data.

In order to examine the difference between development and test set, sentences lengths in German, English and French were measured and shown in tab. 5.13. In the case of German, the length distributions of both data set were visualized in fig. 5.6 and pruned in fig. 5.7.

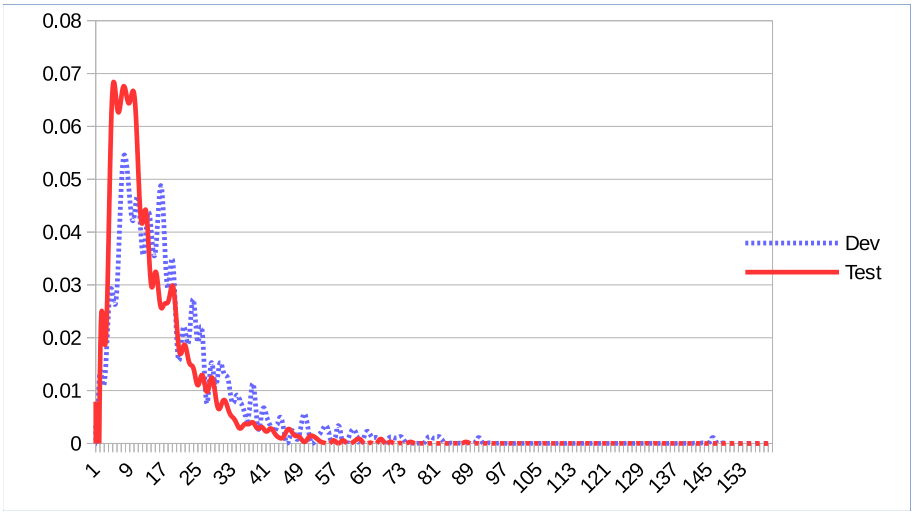


Figure 5.6.: Sentence length distribution of German dev- and test sets

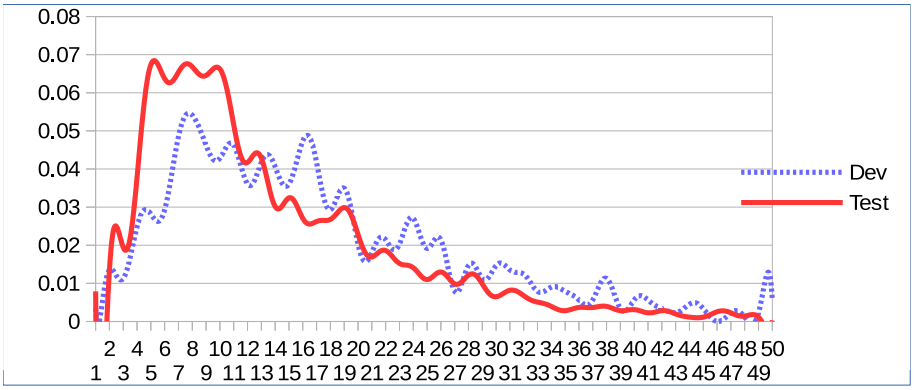


Figure 5.7.: Pruned sentence length distribution of German dev- and test sets

Obviously, in fig. 5.7, there were more shorter sentences in the test set than in development test. The test curve lies under the dev. curve, when considering sentence longer than 10 words. In tab. 5.13, while values of standard variance were quite similar, the averaged sentences in development set were significantly longer (4-5 words), independent of the considered language.

The sentences' length discrepancy could be one of the reasons for poor performance of NNDWLs on development set. Longer input sentences mean there were more context and syntactical information in the structure, which gets lost by considering words only. Imagine that inferring meaning from ("thank", "you") or ("you", "thank") is much easier

than from a word combination consisting of 10 different words, even for human testers. Moreover, the a sentence is, the higher is the probability of dismissing information due to a strict vocabulary (limited by most frequent words). This would additionally let a sentence appear more like noisy data.

Another aspect which led to different translation quality on dev. and test dataset was the amount of sentences. Since the dev. set only possesses 887 sentences, this might be sufficient to be used for hyper-optimizing.

6. Conclusion

6.1. Summary

The aim of this work was to optimize statistical machine translation systems, which are often used when there is no parallel aligned data for a targeted language pair. By transferring the idea of pipelining to the model-level and using an interlingua representation of language, we proposed a multilingual, simple translation model. The model was trained on various language pairs and was capable of translating between languages **without having seen any training data** of that specific pair. Two different network architectures were introduced, referred to as **separated and unseparated multilingual NNDWLs**. They differ in the manner in which the second and penultimate layer of their neural networks were designed. In the case of separate models, the corresponding layers were divided into disjunct sets, each responsible for processing a certain language.

In the evaluation phase, we first compared ourselves to several mono-lingual models: NNDWL as presented in [7], combined with source context from [20], as well as convolutional architectures motivated by [28] and [6]. It turned out that using convolutional architectures for word-tagging tasks in related works led to under-fitting on the DWL-task. Moreover, although the chosen source context resulted in a slight performance improvement (up to +0.25 BLEU), it required more hyper-optimization in order to find a good combination as in [7]. The traditional NNDWL was confirmed as the most suitable model for the multilingual task due to its simplicity while still providing an acceptable performance (+0.2 BLEU on test data).

Afterwards we showed that pipelined translation quality was outperformed by direct translation by approximately 0.6 - 0.7 BLEU points. Furthermore, adapting the pipelining idea to NNDWL in an intuitive way resulted in poorer performance (-0.2 BLEU) compared to the monolingual model, which used the translated output from the previous system directly. Therefore, more accurate multilingual NNDWLs could be applied to reduce this performance gap, or even improve on the monolingual model.

Further experiments were aimed at analyzing the two proposed multilingual architectures. For this purpose, different training procedures were examined. The chosen languages were German (De), English (En) and French (Fr); although the experiments were done on German, English and French only, we would expect similar results for any 3 arbitrary languages. In the first experiment, using two parallel aligned sentence sets from the language pairs *De-En* and *En-Fr*, we confirm that the minimal training data set to allow the network translation from German to French is *De-En*, *En-En* and *En-Fr*. Adding data from additional language pairs (e.g. *De-De* or *Fr-Fr*) resulted in poorer performance according to the intrinsic metrics BCE.

Another experiment, which we referred to as the unmatched corpus experiment, used 2 disjunct data sets *De-En* and *En-Fr*. This actually represents the common use case in which there is no parallel data available for a certain language pair (in this constructed case, De-Fr). Here the proposed multilingual architecture was able to reach the intrinsic translation performance of the unilingual NNDWL (En->Fr), which, unlike the system we propose, requires a preceding complex MT-System for translation from German to French in the background. Moreover, in a normal pivot translation system, multilingual NNDWLs per- formed very similar to unilingual NNDWL (En->Fr). However, when the latter system's translation from English to French was re-optimized on translated output from the first system, multilingual architectures helped to improve translation quality at least on the development data. However, the same behavior could not be observed on the test data.

6.2. Discussion

Although the proposed multilingual architecture sometimes resulted in small improvements regarding translation quality, these results were not statistically stable. In general, all NNDWL models performed very similarly. Analysis suggested that they mostly agreed on the best candidates out of the n-best-list; only low-scored candidates varied strongly dependent on which model was considered. When considering the network's outputs based on the given source sentences, the NNDWLs also seemed to agree about a very small set of the most probable words. It was also observable that the contrast of separate joint models' visualizations was lower than that of their unseparated counterparts, suggesting that they were not good at distinguishing occurring words. Furthermore, the more datasets multilingual models were trained on, the more the candidate ranking distribution differed. Since the translation quality did change when networks used more dataset combinations, adding them could have led them to new search spaces and increased the generalization capabilities of the networks w.r.t. encapsulating implicit meanings.

Another important aspect is that multilingual NNDWL could outperform its monolingual NNDWL counterpart only on the development dataset, not on the test set. On one hand, this could be due to a comparatively small development corpus, which was not sufficient to train optimally generalising models. On the other hand, the discrepancy between these two datasets in terms of sentence length may have played an important role. For all the NNDWLs, the longer the sentence are, the harder the translation taks might be. Training models based on a more difficult and less general validation set could lead to poorly generalizing models.

Finally, although multilingual NNDWLs did not clearly outperform monolingual models, they were still advantageous, since they could translate using a single, much simpler model. Combined with a monolingual language model, it is already possible to generate low-quality translations from given source sentences within a short time frame and on limited hardware. Also, these models are easier and faster to train, deploy and use than a translation pipeline consisting of two complex MT-systems.

6.3. Outlook

The ideas used for the simple NNDWL translation model can be applied to other, more complex models. For example, since convolutional networks for word-tagging tasks seemed underfitted, [11] proposed an improved architecture by exploiting k-max-pooling instead of dismissing too much information by using normal max-pooling. Another possibility would be using a recurrent neural network, which is currently highly preferred in many research fields. Ideally, this would enable translations between many different languages using a single model, without requiring several complex MT-systems. Moreover, such a model could be used to produce acceptable translations within a short time-frame and on limited hardware.

Another aspect is the potential addition of additional data from other languages. In this work, we restricted ourselves to the use of three languages. However, since learning from a larger number of datasets seemed to slightly improve translation quality, the next experiment could be aimed at using even more languages. Ideally, in this way, the neural network models could be forced to derive abstract meaning from sentences, making it more language-independent. Furthermore, we could extend the system to employ non-textual data, adding e.g. video or audio to construct multi-modal inputs.

Bibliography

[1] Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville. "Deep Learning". Book in preparation for MIT Press. 2015. URL: http://www.iro.umontreal.ca/~bengioy/dlbook.

[2] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[3] Mauro Cettolo, Christian Girardi, and Marcello Federico. "WIT^3: Web Inventory of Transcribed and Translated Talks". In: *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*. Trento, Italy, May 2012, pp. 261–268.

[4] David Chiang, Yuval Marton, and Philip Resnik. "Online large-margin training of syntactic and structural translation features". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2008, pp. 224–233.

[5] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. "Implementing neural networks efficiently". In: *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 537–557.

[6] Ronan Collobert and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 160–167.

[7] Thanh-Le Ha, Jan Niehues, and Alex Waibel. "Lexical Translation Model Using a Deep Neural Network Architecture". In: *arXiv preprint arXiv:1504.07395* (2015).

[8] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[9] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[10] Mark Hopkins and Jonathan May. "Tuning as ranking". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2011, pp. 1352–1362.

[11] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. "A convolutional neural network for modelling sentences". In: *arXiv preprint arXiv:1404.2188* (2014).

[12] Philipp Koehn. *Statistical machine translation*. Cambridge University Press, 2009.

[13] Philipp Koehn and Kevin Knight. "Empirical methods for compound splitting". In: *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics. 2003, pp. 187–193.

[14] Philipp Koehn et al. "Moses: Open source toolkit for statistical machine translation". In: *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics. 2007, pp. 177–180.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[16] Lemao Liu et al. "Additive Neural Networks for Statistical Machine Translation." In: *ACL (1)*. 2013, pp. 791–801.

[17] Arne Mauser, Saša Hasan, and Hermann Ney. "Extending statistical machine translation with discriminative and trigger-based lexicon models". In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics. 2009, pp. 210–218.

[18] Jiquan Ngiam et al. "Multimodal deep learning". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 689–696.

[19] Jan Niehues and Muntsin Kolss. "A POS-based model for long-range reorderings in SMT". In: *Proceedings of the Fourth Workshop on Statistical Machine Translation*. Association for Computational Linguistics. 2009, pp. 206–214.

[20] Jan Niehues and Alex Waibel. "An MT error-driven discriminative word lexicon using sentence structure features". In: *Proceedings of the Eighth Workshop on Statistical Machine Translation, Sofia, Bulgaria*. 2013, pp. 512–520.

[21] Jan Niehues et al. "ListNet-based MT Rescoring". In: *EMNLP 2015* (2015), p. 248.

[22] Franz Josef Och. "Minimum error rate training in statistical machine translation". In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics. 2003, pp. 160–167.

[23] Franz Josef Och and Hermann Ney. "A systematic comparison of various statistical alignment models". In: *Computational linguistics* 29.1 (2003), pp. 19–51.

[24] Kishore Papineni et al. "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2002, pp. 311–318.

[25] Kay Rottmann and Stephan Vogel. "Word reordering in statistical machine translation with a POS-based distortion model". In: *Proc. of TMI* (2007), pp. 171–180.

[26] Isabel Slawik et al. "The KIT Translation Systems for IWSLT 2014". In: ().

[27] Vladimir N Vapnik. "An overview of statistical learning theory". In: *Neural Networks, IEEE Transactions on* 10.5 (1999), pp. 988–999.

[28] Alexander Waibel et al. "Phoneme recognition using time-delay neural networks". In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 37.3 (1989), pp. 328–339.

A. Appendix

A.1. Example outputs of multilingual NNDWL

Source	Everybody would like to make people happier.
Reference	Tout le monde veut rendre les autres plus heureux.
NNDWL (De->En->Fr)	gens monde heureux UNK tout les des comme à le de rendre que
NNDWL (En->Fr)	gens UNK les qui personnes le des à de que monde personne faire
U. JointModel	UNK gens les de est qui que des faire comme personnes d c le
U. JointModel + De-De	les gens UNK des le de faire qui la personnes à personne que ont
U. JointModel + De-De, Fr-Fr	gens que faire la des à UNK qui personnes personne aiment rendre
S. JointModel	UNK gens les le que tout personnes pour à d tous de ça faire
S. JointModel + De-De	gens les UNK le de la que ça à personnes faire qui des tout
S. JointModel + De-De, Fr-Fr	UNK de ce les qui la que l à avait il avaient c qu ont le a été en est

Table A.1.: Predicted words by different models, ordered by assigned probabilities (descending order)

A.2. Detailed rescoring results & analysis of multilingual NNDWL

	ListNet		KITPRO		KBMira		MERT	
	Dev	**Test**	**Dev**	**Test**	**Dev**	**Test**	**Dev**	**Test**
Baseline	21.89	18.79	21.23	18.98	22.28	18.68	22.37	18.27
+NNDWL (De->En->Fr)	22.06	18.91	21.54	19.1	22.37	18.68	22.6	18.7
+NNDWL (En->Fr)	21.98	18.88	21.52	19	22.17	18.66	22.5	18.64
+U. JointModel	21.92	18.8	21.44	18.98	22.31	18.76	22.57	18.65
+U. JointModel+De-De	21.97	18.86	21.59	18.97	22.15	18.6	22.57	18.56
+U. JointModel+De-De, Fr-Fr	22.03	18.93	21.64	19.02	22.42	18.79	22.59	18.57
+S. JointModel	21.92	18.8	21.43	18.95	22.21	18.72	22.57	18.65
+S. JointModel+De-De	22.07	18.75	21.46	18.95	22.26	18.74	22.45	18.37
+S. JointModel+De-De, Fr-Fr	21.94	18.88	21.4	18.98	22.2	18.8	22.49	18.77
+UniformNoise	21.92	18.71	21.58	18.93	21.13	18.11	22.44	18.48

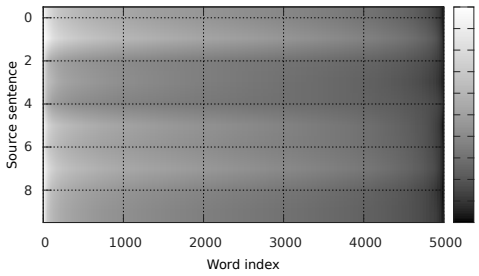
Table A.2.: Translation quality of multi- and unilingual NNDWLs in addition to De-En-Fr-systems (unmatched corpus)

	ListNet		KITPRO		KBMira		MERT	
	Dev	**Test**	**Dev**	**Test**	**Dev**	**Test**	**Dev**	**Test**
Baseline	14.52	18.73	14.39	18.85	14.74	18.79	14.98	18.5
+NNDWL (De->En->Fr)	14.82	18.85	14.53	18.9	14.81	18.98	15.04	18.58
+NNDWL (En->Fr)	14.71	18.66	14.57	18.79	14.86	18.8	15.09	18.58
+U. JointModel	14.85	18.55	14.55	18.74	14.92	18.48	15.1	18.59
+U. JointModel+De-De	14.81	18.66	14.51	18.88	14.99	18.75	15.19	18.28
+U. JointModel+De-De, Fr-Fr	14.81	18.63	14.64	18.88	14.8	18.65	15.11	18.39
+S. JointModel	14.85	18.55	14.53	18.79	14.9	18.62	15.11	18.39
+S. JointModel+De-De	14.68	18.57	14.5	18.83	14.86	18.84	15.05	18.19
+S. JointModel+De-De, Fr-Fr	14.76	18.72	14.55	18.91	14.91	18.83	15.1	18.59
+UniformNoise	14.63	18.56	14.3	18.86	13.76	16.04	14.95	18.46

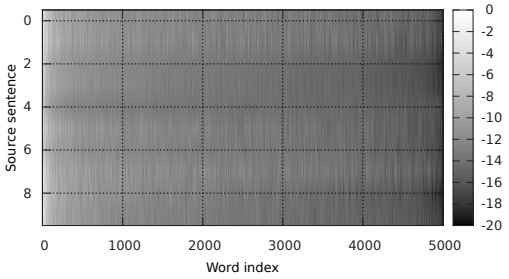
Table A.3.: Translation quality of multi- and unilingual NNDWLs in addition to reoptimized De-En-Fr-systems (unmatched corpus)

	ListNet		KITPRO		KBMira		MERT	
	Dev	**Test**	**Dev**	**Test**	**Dev**	**Test**	**Dev**	**Test**
Baseline	24.27	20.77	23.78	21.08	24.67	20.62	24.87	20.6
+NNDWL (De->En->Fr)	24.27	21.04	23.99	21.22	24.7	20.76	24.93	20.67
+NNDWL (En->Fr)	24.16	21.06	23.81	21.17	24.64	20.72	24.91	20.76
+U. JointModel+De-De	24.07	20.99	23.89	21.06	24.65	20.77	24.79	20.65
+S. JointModel	24.25	21.02	23.84	21.15	24.64	20.83	24.87	20.6
+S. JointModel+ De-De	24.16	20.98	23.93	21.09	24.68	20.7	24.91	20.62
+S. JointModel+De-De, Fr-Fr	24.13	20.98	23.95	21.15	24.66	20.7	24.91	20.69

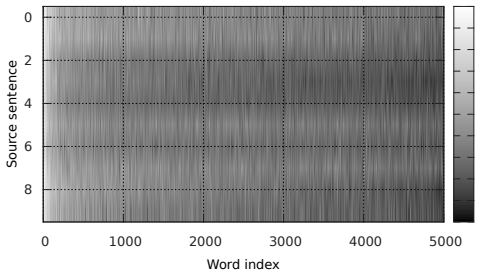
Table A.4.: Translation quality of multi- and unilingual NNDWLs in addition to De-En-Fr-systems (matched corpus)



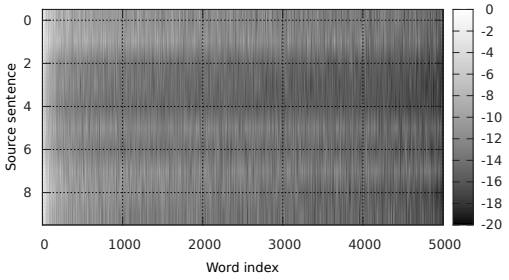
(a) NNDWL (De->En->Fr)



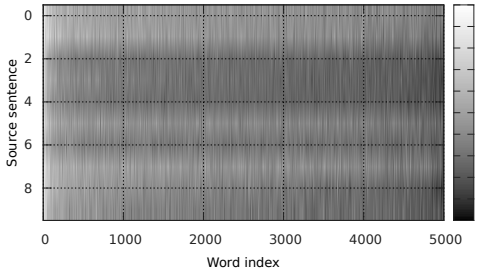
(b) NNDWL (En->Fr)



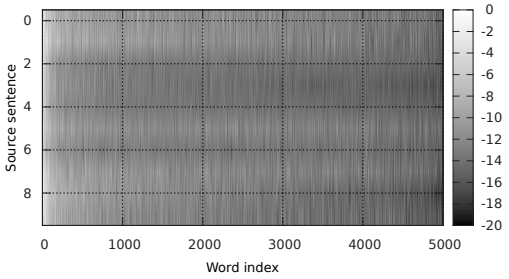
(c) U. JointModel



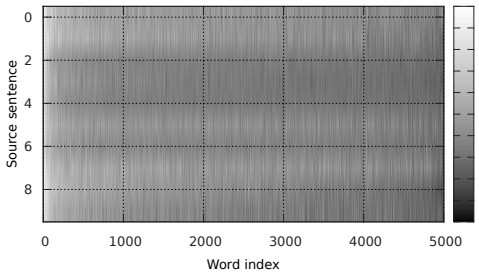
(d) U. JointModel + De-De



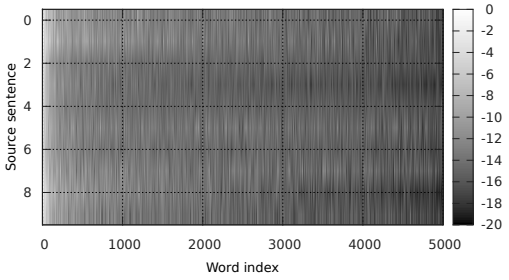
(e) U. JointModel + De-De, Fr-Fr



(f) S. JointModel

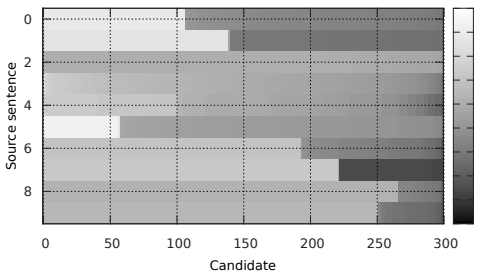


(g) S. JointModel + De-De

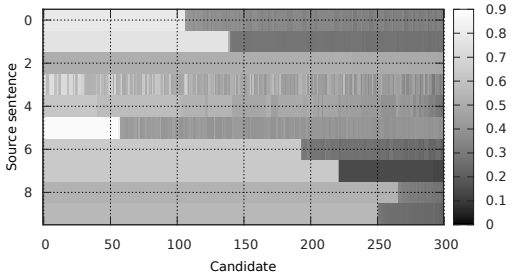


(h) S. JointModel + De-De, Fr-Fr

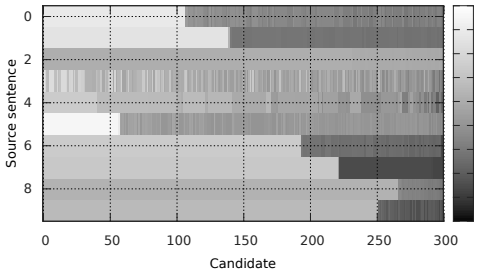
Figure A.1.: NNDWL computed French words' probabilities , given the **10** sentences from development data set



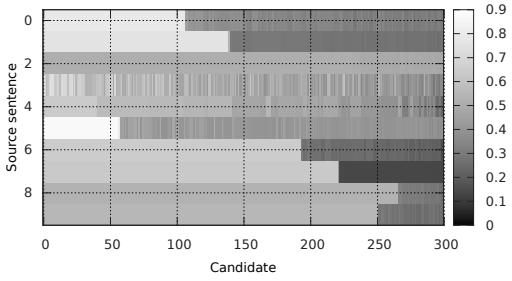
(a) NNDWL (De->En->Fr)



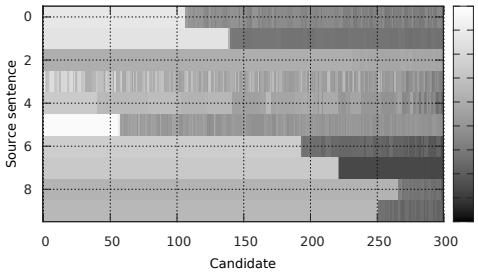
(b) NNDWL (En->Fr)



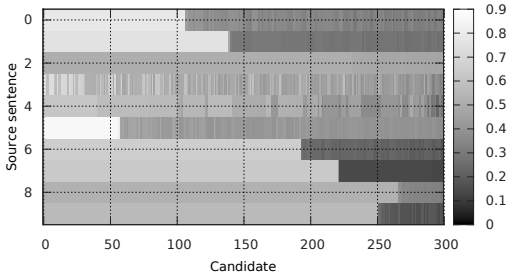
(c) U. JointModel



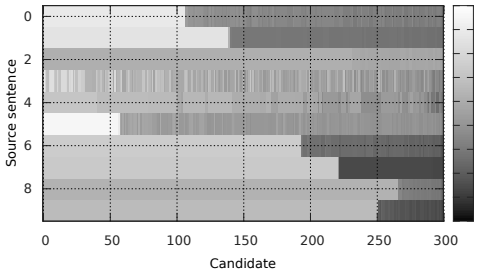
(d) U. JointModel + De-De



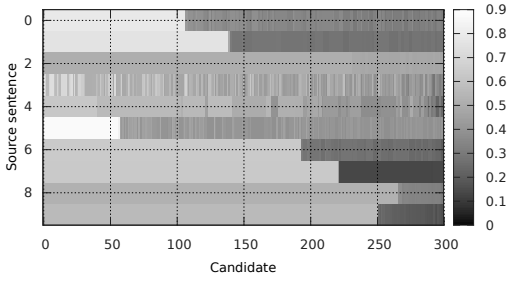
(e) U. JointModel + De-De, Fr-Fr



(f) S. JointModel

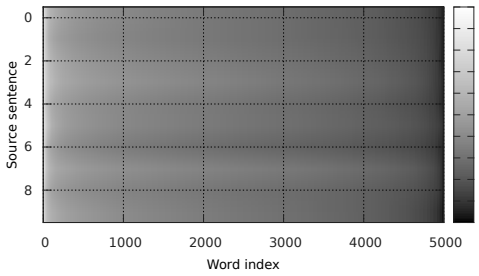


(g) S. JointModel + De-De

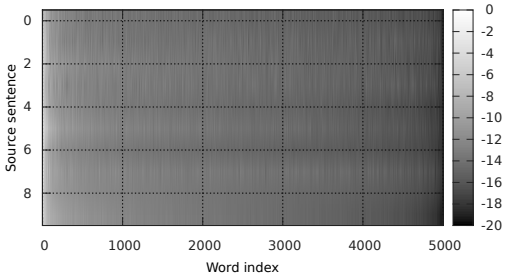


(h) S. JointModel + De-De, Fr-Fr

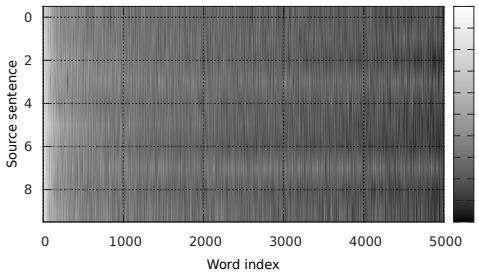
Figure A.2.: NNDWL scored candidates for the first **10** sentences of development data set



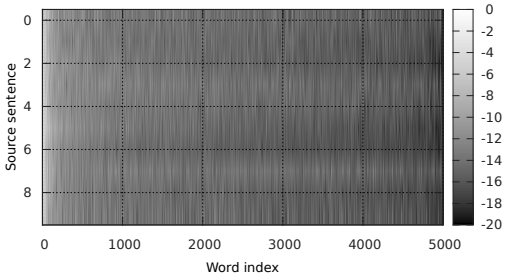
(a) NNDWL (De->En->Fr)



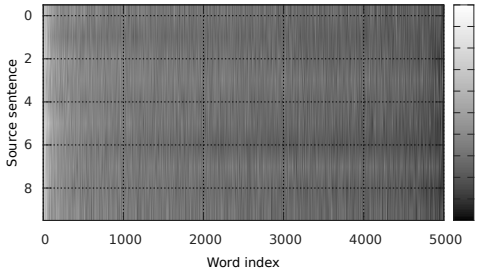
(b) NNDWL (En->Fr)



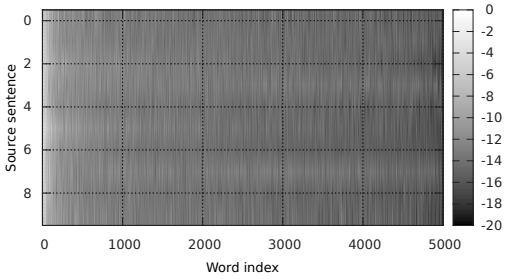
(c) U. JointModel



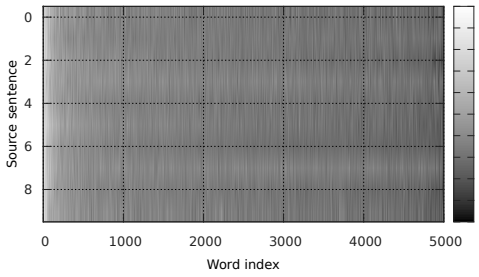
(d) U. JointModel + De-De



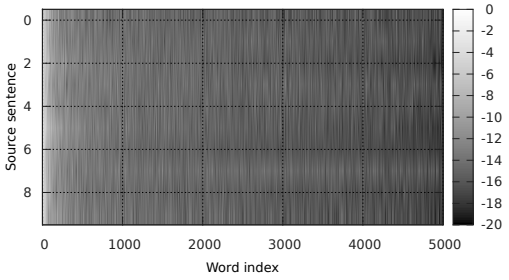
(e) U. JointModel + De-De, Fr-Fr



(f) S. JointModel

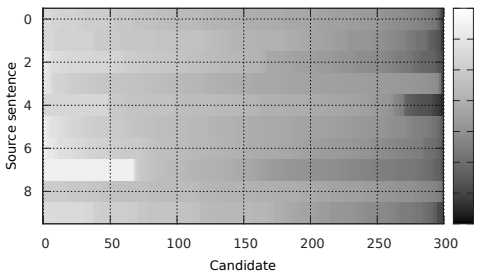


(g) S. JointModel + De-De

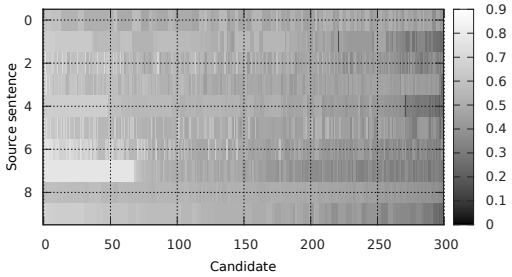


(h) S. JointModel + De-De, Fr-Fr

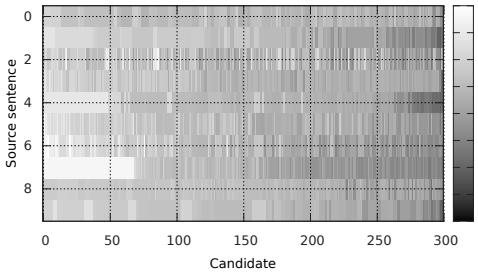
Figure A.3.: NNDWL computed French words' probabilities , given the **10** sentences from test data set



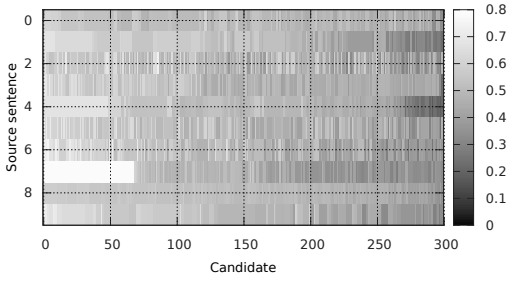
(a) NNDWL (De->En->Fr)



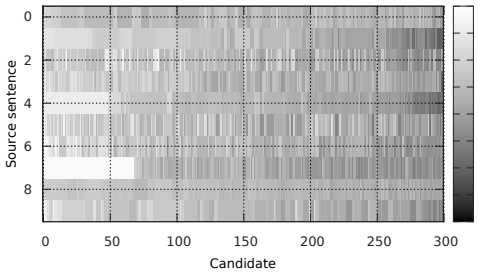
(b) NNDWL (En->Fr)



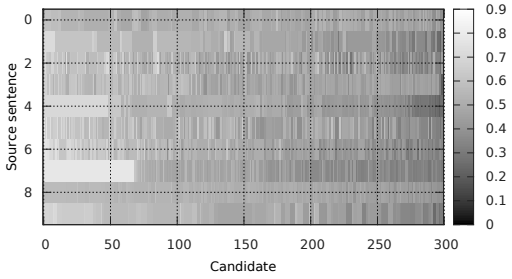
(c) U. JointModel



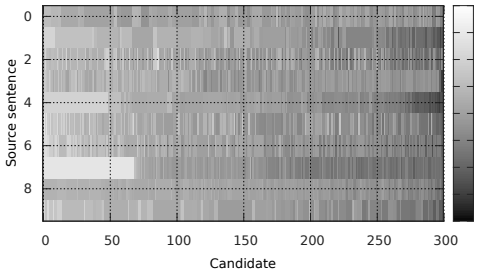
(d) U. JointModel + De-De



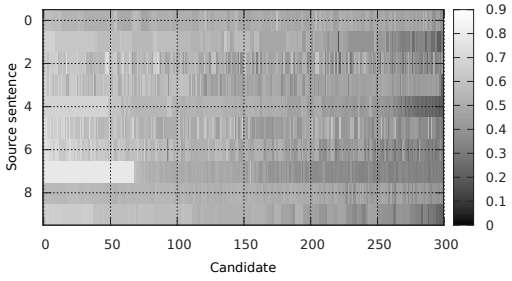
(e) U. JointModel + De-De, Fr-Fr



(f) S. JointModel

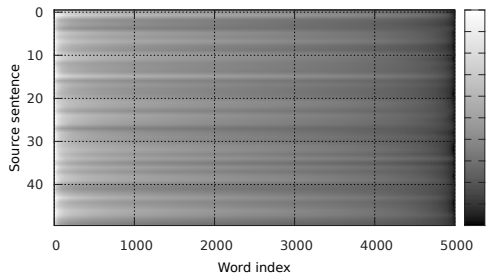


(g) S. JointModel + De-De

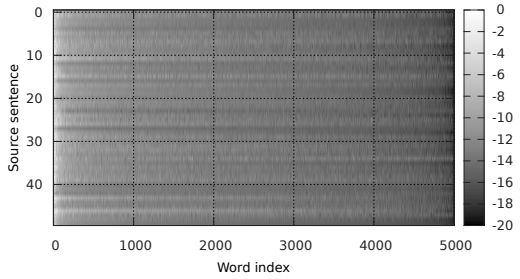


(h) S. JointModel + De-De, Fr-Fr

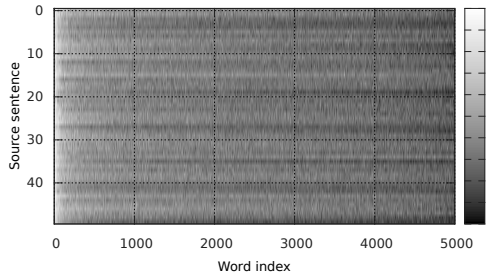
Figure A.4.: NNDWL scored candidates for the first **10** sentences of test data set



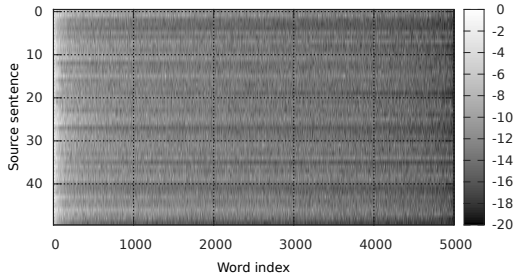
(a) NNDWL (De->En->Fr)



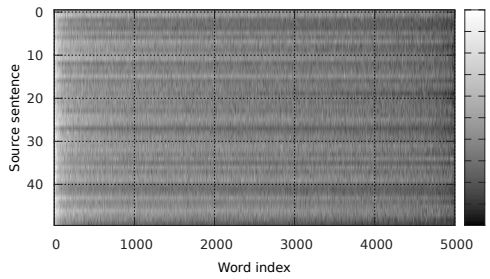
(b) NNDWL (En->Fr)



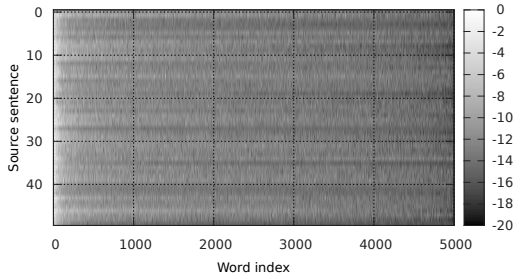
(c) U. JointModel



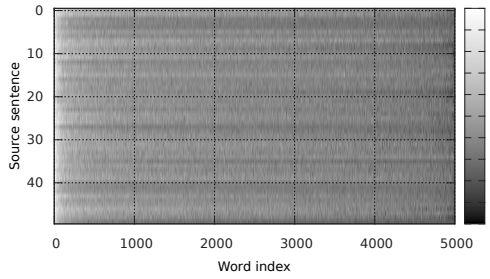
(d) U. JointModel + De-De



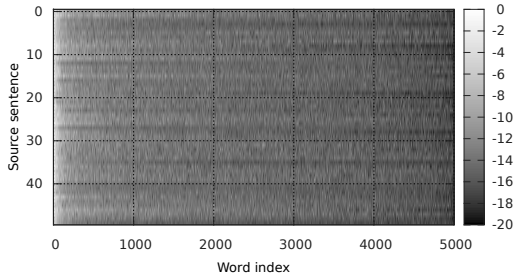
(e) U. JointModel + De-De, Fr-Fr



(f) S. JointModel

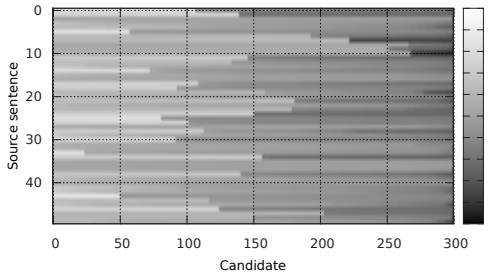


(g) S. JointModel + De-De

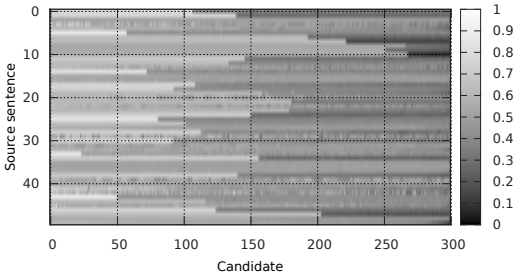


(h) S. JointModel + De-De, Fr-Fr

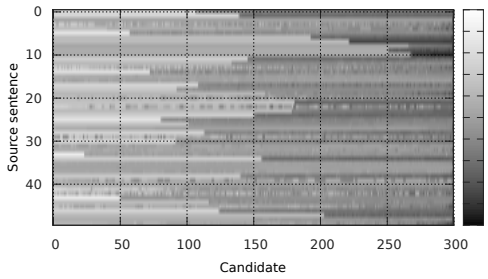
Figure A.5.: NNDWL computed French words' probabilities , given the **50** sentences from development data set



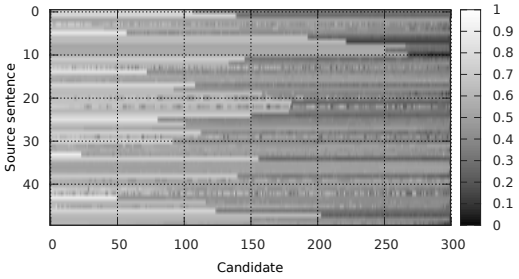
(a) NNDWL (De->En->Fr)



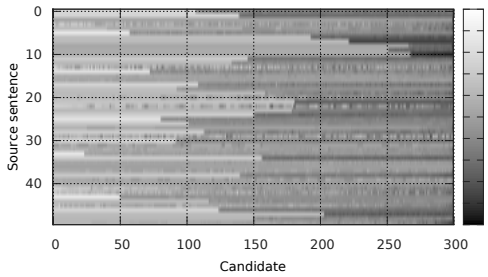
(b) NNDWL (En->Fr)



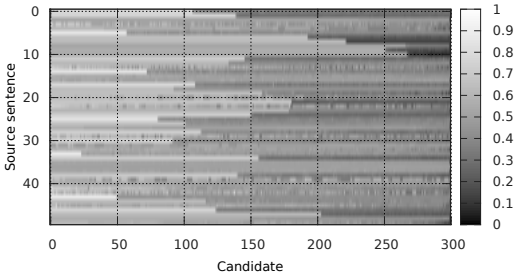
(c) U. JointModel



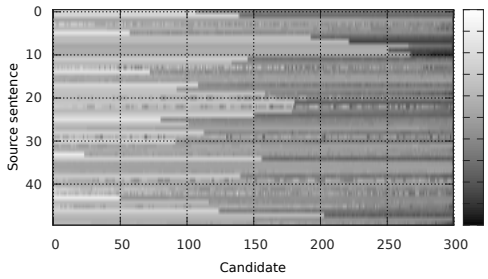
(d) U. JointModel + De-De



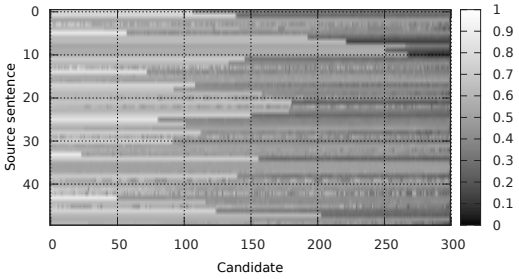
(e) U. JointModel + De-De, Fr-Fr



(f) S. JointModel

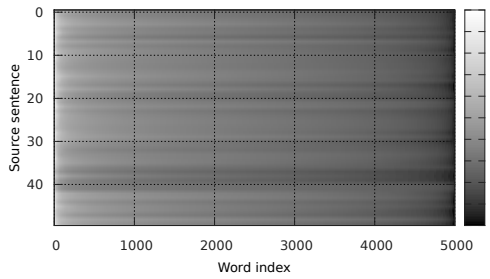


(g) S. JointModel + De-De

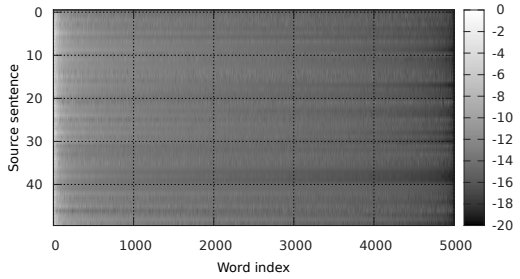


(h) S. JointModel + De-De, Fr-Fr

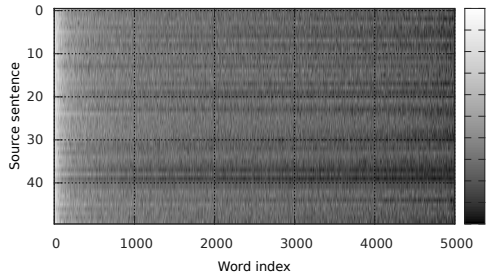
Figure A.6.: NNDWL scored candidates for the first **50** sentences of development data set



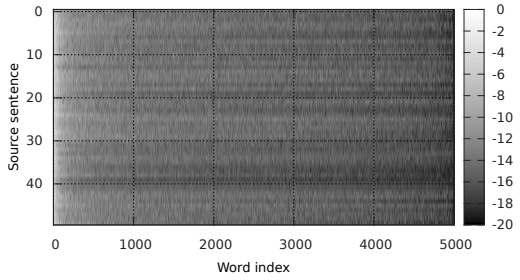
(a) NNDWL (De->En->Fr)



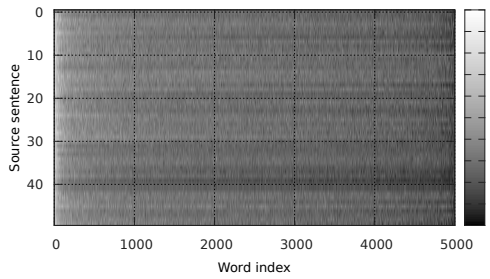
(b) NNDWL (En->Fr)



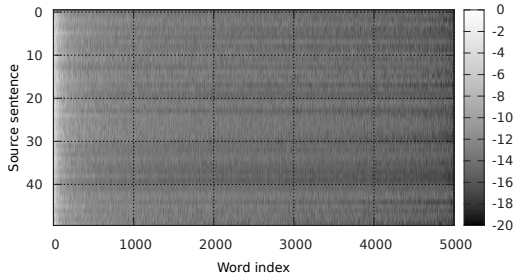
(c) U. JointModel



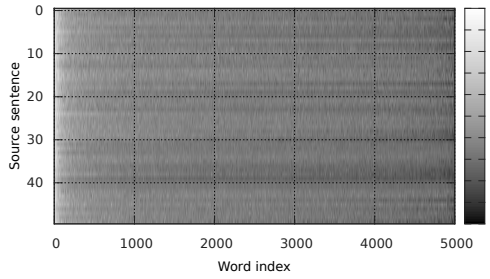
(d) U. JointModel + De-De



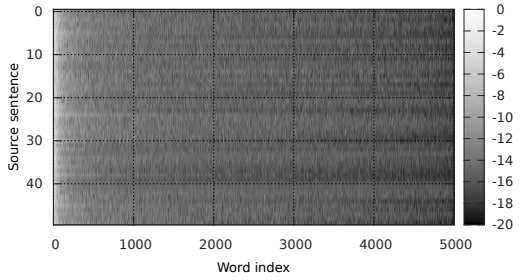
(e) U. JointModel + De-De, Fr-Fr



(f) S. JointModel

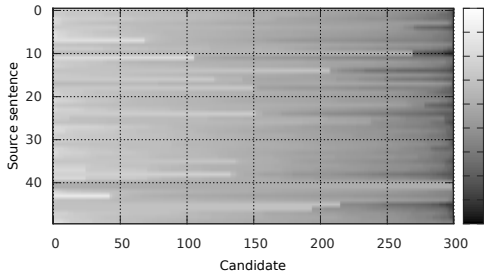


(g) S. JointModel + De-De

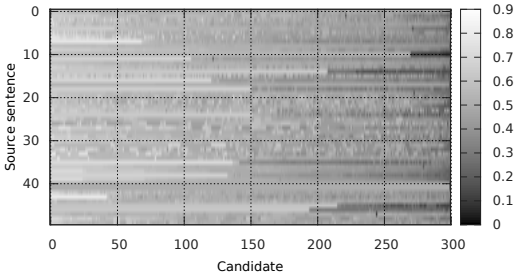


(h) S. JointModel + De-De, Fr-Fr

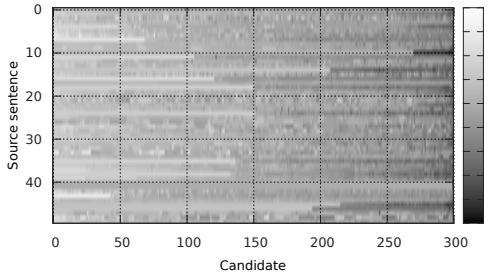
Figure A.7.: NNDWL computed French words' probabilities , given the **50** sentences from test data set



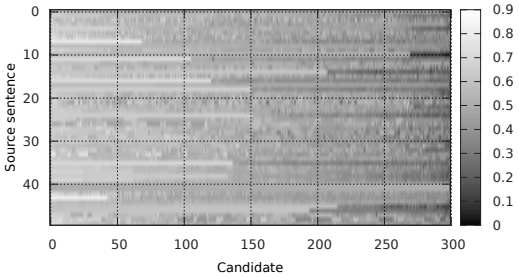
(a) NNDWL (De->En->Fr)



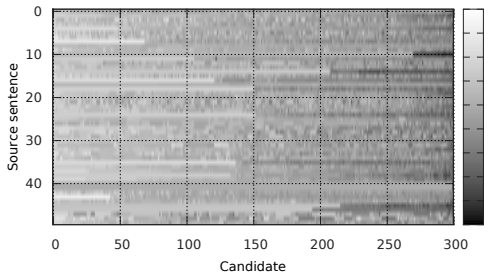
(b) NNDWL (En->Fr)



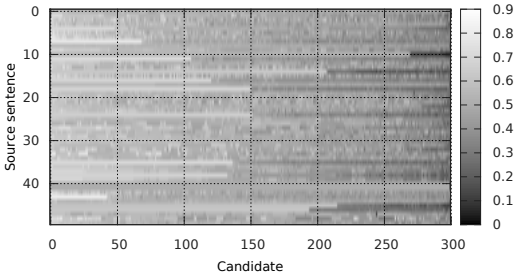
(c) U. JointModel



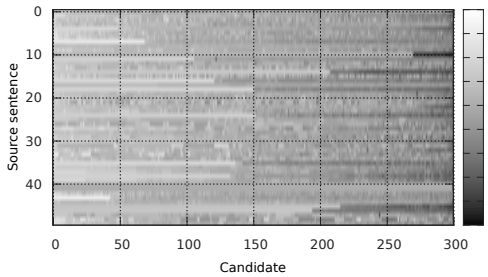
(d) U. JointModel + De-De



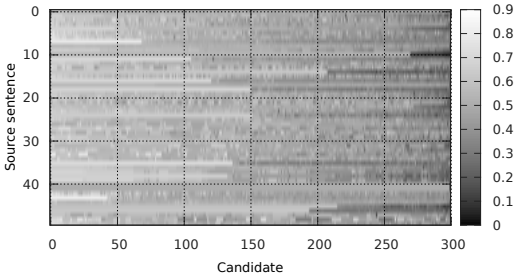
(e) U. JointModel + De-De, Fr-Fr



(f) S. JointModel



(g) S. JointModel + De-De



(h) S. JointModel + De-De, Fr-Fr

Figure A.8.: NNDWL scored candidates for the first **50** sentences of test data set

	ListNet		KITPRO		KBMira		MERT	
	Dev	**Test**	**Dev**	**Test**	**Dev**	**Test**	**Dev**	**Test**
Baseline	16.69	20.62	16.45	20.96	16.86	20.66	17.29	20.63
+NNDWL (De->En->Fr)	16.82	20.81	16.49	21.12	17.08	20.67	17.3	20.73
+NNDWL (En->Fr)	16.77	20.85	16.57	21.09	16.92	20.65	17.25	20.74
+U. JointModel+De-De	16.83	20.79	16.55	21.09	17.09	20.76	17.28	20.66
+S. JointModel	16.77	20.81	16.55	21.05	16.99	20.72	17.28	20.91
+S. JointModel+ De-De	16.85	20.82	16.57	21.07	16.92	20.78	17.28	20.9
+S. JointModel+De-De, Fr-Fr	16.83	20.79	16.53	21.09	17.03	20.6	17.28	20.71

Table A.5.: Translation quality of multi- and unilingual NNDWLs in addition to reoptimized De-En-Fr-systems (matched corpus)