UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Quality Estimation from Scratch

Master's Thesis
March 22, 2016

Julia Kreutzer
kreutzer@cl.uni-heidelberg.de

Institut für Computerlinguistik
Universität Heidelberg

Supervisor Prof. Dr. Stefan Riezler
Reviewers Prof. Dr. Stefan Riezler
Dr. Artem Sokolov

Abstract

This thesis presents a deep neural network for word-level machine translation quality estimation. The model extends the feedforward multi-layer architecture by [Collobert *et al.*, 2011] to learning continuous space representations for bilingual contexts from scratch. By means of stochastic gradient descent and backpropagation of errors, the model is trained for binary classification of translated words, given only the source sentence and the machine translation. We enhance this model with alignments, and unsupervised pre-training of word representations allows for leveraging large monolingual corpora for supervised quality estimation training. Evaluating it on the data provided by the Workshop on Statistical Machine Translation 2014 and 2015, the model yields competitive results across languages and datasets. A linear combination of the deep model and a shallow linear model trained on baseline features further improves over both individual models. Furthermore, the bilingual word representations learnt during supervised training for quality estimation prove useful for other cross-lingual tasks.

Zusammenfassung

Diese Masterarbeit stellt ein Modell für die Einschätzung der Qualität maschineller Übersetzungen ("Quality Estimation") auf der Wortebene vor, das auf einem künstlichen neuronalen Netz basiert. Dieses Modell erweitert das "Multi-Layer Perceptron" von [Collobert *et al.*, 2011], um aus bilingualen Kontexten geeignete Vektorrepräsentationen zu lernen. Als Eingabe erhält das Modell den Satz in der Quellsprache und die maschinelle Übersetzung. Mithilfe des Gradientenabstiegsverfahrens und der Rückpropagierung von Fehlern in der binären Klassifikation wird das Modell trainiert. Das Modell profitiert außerdem von Wortalignierungen und unüberwacht trainierten Wortrepräsentationen, die es erlauben, umfangreiche einsprachige Korpora für das überwachte Training der Quality Estimation auszunutzen. Angewandt auf die Daten, die im Rahmen der Workshops on Statistical Machine Translation 2014 und 2015 bereitgestellt werden, erzielt das Modell sprach- und datenübergreifend sehr gute Ergebnisse. Eine lineare Kombination dieses nichtlinearen Klassifizierers mit einem linearen Klassifizierer, der auf Standard-Features trainiert wird, führt zu einer Systemkombination, die bessere Vorhersagen erlaubt als beide einzelnen Modelle. Darüber hinaus stellen wir fest, dass die gelernten bilingualen Wortrepräsentationen für weitere sprachübergreifende Aufgaben sinnvoll eingesetzt werden können.

Contents

List of Figures VI

List of Tables VII

1 Introduction 1

2 Quality Estimation 3
- 2.1 The Task and its Challenges . . . 3
- 2.2 Recent Approaches . . . 7

3 Deep Learning for NLP 9
- 3.1 Deep Neural Networks for NLP . . . 9
- 3.2 Learning Distributed Word Representations . . . 11
- 3.3 SENNA: NLP "Almost from Scratch" . . . 16
- 3.4 Deep Learning for QE . . . 20

4 Quality Estimation from Scratch 21
- 4.1 Neural Network Architecture . . . 21
- 4.2 Training . . . 26
- 4.3 Unsupervised Pre-training . . . 28
- 4.4 Baseline Features and System Combination . . . 28

5 Experiments 31
- 5.1 WMT QE Tasks . . . 31
- 5.2 Additional Experiments . . . 39
- 5.3 Evaluation of the Word Embeddings . . . 44

6 Discussion 50

7 Conclusion 53

A Appendix 54
A.1 Gradients 54
A.2 Word Embedding Visualizations 57

Bibliography 59

List of Figures

1 SENNA neural network architecture for the window approach. Source: [Collobert *et al.*, 2011] . . . 16
2 SENNA neural network architecture for the sentence approach. Source [Collobert *et al.*, 2011] . . . 17
3 QUETCH architecture . . . 22
4 QUETCH+ architecture . . . 22
5 Accuracy learning curves . . . 40
6 BAD F1 learning curves . . . 40
7 Visualized `word2vec` embeddings . . . 57
8 Visualized word embeddings of "late" QUETCH model . . . 57
9 Visualized word embeddings of submitted QUETCH model . . . 58
10 Visualized word embeddings of vanilla model . . . 58

List of Tables

1 WMT QE scores for three levels of granularity 5
2 Common QE features 8

3 Confusion matrix for OK-classification. 31
4 Confusion matrix for BAD-classification. 31
5 WMT14 example 32
6 WMT14 label distribution 33
7 WMT14 QUETCH results 34
8 WMT14 multilingual results 35
9 WMT14 winner results 35
10 WMT15 baseline features 36
11 WMT15 examples 37
12 WMT15 label distribution 37
13 WMT15 QUETCH and QUETCH+ results 38
14 WMT15 winner results 38
15 WMT14 sentence-level QUETCH results 41
16 WMT14 sentence-level winner results 42
17 More QUETCH results 43
18 Nearest neighbours for `word2vec` and QUETCH embeddings 48
19 Monolingual word similarity results 49
20 Cross-lingual word similarity results 49
21 Monolingual document classification results 49
22 Cross-lingual document classification results 49

1 Introduction

How good is machine translation? – While there is probably no universal answer to this question, it is crucially important to anyone who uses machine translated texts. Whether it its used to translate scientific publications, translate product descriptions or websites – the user needs to know how reliable these automatic translations are, how severe translation errors are, and whether they need further post-editing.

When developing a machine translation (MT) system, evaluation metrics are based on the comparison to human reference translations, answering the question "How close is machine translation to human translation?". In contrast, quality estimation (QE) is the task to evaluate machine translations without human reference translations and without knowledge about the MT system that generated it.
QE can be assessed on various levels of granularity. At sentence-level, a QE system could e.g. rank the translation quality on a scale from 0 to 5, on word-level it could e.g. tell for each word whether it was well translated or which types of errors are prevelant. With this information, human or automatic post-editing can be done in a more efficient way, since correct parts of the text can be skipped and post-editing can focus on the relevant, incorrect parts.

This work presents a quality estimation system built on a deep neural network, as a bilingual extension of the framework proposed by [Collobert *et al.*, 2011] that addresses various natural language processing (NLP) classification tasks. One of the main advantages of such deep models is that they allow learning "from scratch", i.e. from raw input like words or letters, replacing task-specific feature engineering. Winning the official Workshop of Statistical Machine Translation (WMT) 2015 QE task, our deep learning QE system in combination with a baseline system was proven sucessful for QE on word-level.

The remainder of this thesis is organized as follows. First we address quality estimation as a machine learning task and present recent approaches. Section 3 then gives a rough overview over methods for deep machine learning for NLP. Both the task (QE) and the method (deep machine learning) are united in Section 4, where the novel approach of this thesis is presented in detail. Experiments on WMT data and an inspection of the learned parameters facilitate an evaluation of this approach in Section 5. Subsequently, the discussion in Section 6 identifies general problems

of the QE task and critically examines the presented approach. A summary and ideas for further work conclude this thesis with Section 7.
Note that the proposed models were already published in [Kreutzer *et al.*, 2015] within the WMT15 evaluation. This thesis further elaborates the motivation, provides precise model definitions, compares to other approaches, and presents additional experiments.

2 Quality Estimation

2.1 The Task and its Challenges

From SMT to QE: Measuring Translation Quality

As sketched in the introduction, the task of quality estimation evolved from the insuffiency of machine translation quality. Statistical machine translation systems have significantly improved over the last decade [Graham *et al.*, 2014]. When developing and comparing SMT systems, their performance is most prominently evaluated by means of the BLEU score [Papineni *et al.*, 2002]. The BLEU score is a corpus-based metric that measures how precisely the translation matches one or more human reference translations. Most widely used as a sentence-based machine translation evaluation metric is METEOR [Banerjee & Lavie, 2005] that extends the capabilities of BLEU by synonym matching and stemming. Both BLEU and METEOR have been evaluated against human judgment, and finding appropriate metrics is still subject to active research (e.g. in the WMT metrics shared task). What characterizes all these evaluation metrics is the strategy to evaluate a machine translation in terms of differences to a human reference translation. The judgment of translation quality is therefore highly dependent on the characteristics of the reference and the technique for comparison. Ideally, more than one reference translation exists for each sentence in the parallel training data, since translation is never a task with only one correct solution. References can be diverse and are costly to obtain for large corpora. But they are the essence to good SMT if considered as a supervised task.

The idea behind quality estimation, however, is to part judgment on translation quality from the comparison with reference translations. It is assumed that the indicators for translation quality are immanent in the source and target text. One motivation is that humans can judge on translation quality without being offered a reference translation [Temnikova *et al.*, 2015]. This judgment is naturally oriented towards errors, i.e. instead of first manually translating the full sentence and then comparing it to the reference, judgment is rather led by the question "How many and which errors with which severity are contained in the translation?". [Specia *et al.*, 2010] demonstrate that the correlation with human annotations for post-edit effort is higher for sentence-based QE scores predicted by a SVM regression model than BLEU and METEOR scores on English to

Danish and Spanish translations. These findings support the idea that QE predictions can come closer to human judgment than comparisons to references.

Metrics for QE

As mentioned in Section 1, the development of methods for QE is mainly driven by the annual QE task of the WMT, where labeled training and test data are provided for supervised learning. Therefore, the task of QE has most prominently been understood as a supervised task for machine learning. Starting with sentence-level QE in 2012's WMT, QE was extended to word (WMT13) and paragraph level (WMT15). There are two variations of the tasks: scoring and ranking. Scoring aims to predict correct quality scores or labels, whereas ranking avoids these explicit scores and aims to find a ranking of multiple translations according to their quality.
For each level, several scoring techniques and gold label sets with varying granularities were developed, as summarized in Table 1. On the paragraph level only METEOR has been used for scoring. On the sentence level three different types of metrics have been considered:

- Human Targeted Error Rate (HTER) [Snover *et al.*, 2006] that measures the edit-distance between the machine translation and a human post-edition (WMT14.1.2, WMT15.1)
- Scores for perceived post edit effort in a range from 1 to 3 (WMT14.1.1)
- The amount of required post-editing time in milliseconds (WMT14.1.3)

On the word level both scores based on post-editing and labels based on human-designed categories were developed:

- Edit-action as labels: good/delete/substitute (WMT13)
- Scores for perceived post-edit effort on a likert scale from 1 to 5 (WMT12)
- The 20 Multidimensional Quality Metrics (MQM) [Uszkoreit & Lommel, 2013] core error categories: Terminology, Mistranslation, Omission, Addition, Untranslated, Accuracy, Style/Register, Capitalization, Spelling, Punctuation, Typography, Morphology (Word Form), Part of Speech, Agreement, Word Order, Function Words, Tense/Aspect/Mood, Grammar, Unintelligible, Fluency, or OK (WMT14 multi-class)
- The MQM level-1 metrics: good/accuracy/fluency (WMT14)
- Binary labels: OK/BAD

	Paragraph Level	**Sentence Level**	**Word Level**
Distance to a post-edition	METEOR		
		HTER	HTER
Post-edit effort		perceived PE effort [1,3]	perceived PE effort [1,5]
		PE time in ms	
			good/delete/substitute
Quality labels			core MQM
			Level-1 MQM
			OK/BAD

Table 1: WMT QE scores for three levels of granularity

The choice of gold labels or scores for quality estimation is associated with the perspectives on quality estimation as a task. Predicting e.g. post-editing times emphasizes the goal to make post-editing more efficient, whereas predicting MQM categories expresses the objective to statistically capture the linguistic characteristics of machine translation errors. It is notable that the sentence- and phrase-level WMT evaluations appear to constitute a understanding of quality oriented towards post-editions ("How much edits are needed for this translation to turn it into a good translation?"), whereas word-level evaluations furthermore consider the notion of intrinsic translation quality ("Is this a good translation? Which parts are bad?").

This work focuses on word-level QE, since it is the finest-grained task, and document- and sentence-level QE might be understood as a composition of word-level QE. However, this has not been empirically explored or validated, at least [Specia *et al.*, 2015] found that using word-level oracle labels improves the prediction on sentence-level, whereas there was no significant improvement over the baseline when using sentence-level predictions for document-level prediction. Also, [de Souza *et al.*, 2014] used binary predictions as features for finer-grained predictions successfully in the WMT14 evaluation. But so far, the tasks have mostly been treated independently, which is also due to their different evaluation metrics as explained above.

Besides the application for post-editing sketched in the introduction, QE can be applied for the following purposes:

- re-rank translation candidates of an SMT decoder or in spoken language translation [Ng *et al.*, 2015]
- select the best translation among options from multiple machine translation systems [Shah & Specia, 2014]

- filter pairs of sentences from a weakly-parallel corpus that are most likely to be translations and use them for SMT training
- inform readers of the target language to which extent they can rely on a translation
- help decide whether a given translation is good enough for publishing as is [Soricut & Echihabi, 2010]
- for post-editing: highlighting words and phrases that need revision [Bach *et al.*, 2011]
- for computer-aided translation (CAT): present binary scores to guide the post-editor to either post-edit or re-translate [Turchi *et al.*, 2015]

Although these application are reasonable in principle, only some of them have already proven to be realizable in real-world scenarios. [Turchi *et al.*, 2015] found small but statistically evidence that QE labels can actually lead to productivity gains, but only under specific conditions. As [Bojar *et al.*, 2014] conclude, it remains questionable whether current QE systems are already precise enough to be employed in these practical settings. So for now the primary goal is to improve QE techniques such that they will be one day good enough to measurably improve SMT systems, post-editing and CAT. The fact that most of the WMT14 submissions for the word-level task did not beat a trivial baseline might seem striking, so the next paragraph explains the challenges of word-level QE in detail.

Why is QE challenging?

There are several reasons that make QE a challenging task:

1. The predictions are MT-system independent, i.e. the MT system that created the translations is treated as a black box. Although one could train a QE model for one specific MT system to adapt the model to the typical errors of this very system, QE aims to generalize over specific MT systems to be useful for system comparison. Also, this is an elementary difference between the goals of quality estimation and SMT confidence estimation [Ueffing *et al.*, 2003].
2. Like MT, it is a bilingual task, so it requires machine accessible knowledge about the source language, the target language, and the interaction of both. The way to bring together these bits of information is challenging, as well as finding appropriate representations for this knowledge as such.

3. The datasets to train QE models on are rather small, since manual annotation is costly. For multi-label error classification sparsity issues occur, as e.g. in the WMT14 evaluation, where the number of samples for some of the MQM core category is very low.

4. The datasets are skewed, so usually there are more samples for correct translations than for incorrect ones. This shows that SMT is at a stage where most of its output can be relied on, but corrections are still needed.

The first two challenges originate from the task itself, whereas the two latter challenges result from the way QE was so far realized as a supervised classification task in WMT. When designing a QE system, one has to take these challenges into account. This requires a careful selection of features and a suitable of machine learning classifier. Therefore, the next section will explore how the task of word-level QE has been addressed so far, before finally introducing our novel approach for a deep learning QE classifier.

2.2 Recent Approaches

Most commonly, word-level QE has been comprehended as a standard classification task, where instances are single target words, and labels depend on the chosen granularity. This is why common evaluation metrics like recall, precision, F-measure, and accuracy seem reasonable. These have been used in the WMT QE task, so a comparison can be made with respect to these metrics. As in other word tagging tasks (like e.g. word-level sentiment classification), the primary focus in recent work has been on the creation and selection of suitable features, whereas the range of classifiers applied is comparatively low (prevalently Support Vector Machines (SVM) and Conditional Random Fields (CRF)). A roughly categorized overview of common word-level QE feature sets is presented in Table 2. The variety of these features is high: They range from simple textual features like the target word with its context, over features that require linguistic information like part-of-speech (POS) tags, to complex features like semantic similarity or resource-based features like language models. Some of them are designed to capture the characteristics of the target (e.g. target language model), while others aim to capture the association between source and target (e.g. pseudo-references). Naturally, the effectiveness of these features is dependent on the language pair, the domain of the translations, the the granularity of the labels, and the choice of classifier and evaluation metric. Although we cannot make global jugdments on their effectiveness (see [Shah *et al.*, 2013] for a detailed investigation), we have the impression that in particular language model probabilities, alignment information and pseudo-reference

Category	Feature Set	Reference
Lexical information	lexical categories (stopword, NE, proper name, numerical)	[Luong *et al.*, 2014]
	word lexicons	[de Souza *et al.*, 2014]
	POS tag	[de Souza *et al.*, 2014]
Fluency	target language model probability	[Wisniewski *et al.*, 2014]
	POS tag language model	[Hokamp *et al.*, 2014]
	stopword language model	[Hokamp *et al.*, 2014]
Semantics	semantic similarity	[Hokamp *et al.*, 2014]
	target polysemy count	[Luong *et al.*, 2014]
Surface	word length, location, prefix, suffix, form, context	[Biçici & Way, 2014]
Syntax	constituent label	[Luong *et al.*, 2014]
	depth in parse tree	[Luong *et al.*, 2014]
Error	error grammar parsing	[Hokamp *et al.*, 2014]
	probability that a training word is BAD	[Wisniewski *et al.*, 2014]
MT	pseudo-references	[Wisniewski *et al.*, 2014, Sánchez & Forcada, 2015]
	confusion networks descriptors computed over translation hypotheses	[de Souza *et al.*, 2014]
	word posterior probabilities extracted from n-best lists	[de Souza *et al.*, 2014]

Table 2: Common feature sets for word-level QE

features are most popularly used, probably because they were proven essential for SMT[1]. The open-source QE framework QUEST [Specia *et al.*, 2013], that was also used for baseline feature extraction in WMT, provides a set of currently 111 different black-box features for word-level QE.

Concerning the problem of the skewed training data, [Shang *et al.*, 2015] present two strategies to re-balance the datasets: One approach is to introduce sub-labels for OK, another is to delete completely correct sentences from training set and replace them with duplicated sentences that have a high BAD ratio. [Shah *et al.*, 2015b] pursue a similar filtering approach and reduce the dataset to sentences that have a high proportion of BAD words. However, both approaches do not present a universal solution of how to exploit all labeled data in their original form.

1 Pseudo-references are not used in SMT in the exact same way as for QE. But the principle is the same: SMT models with the goal to maximize BLEU are trained to match references; QE models are trained with the information where the translation already matches the references.

3 Deep Learning for NLP

After having familiarized the reader with the task of QE, we now continue with an introduction to deep learning of neural networks in the field of NLP.

3.1 Deep Neural Networks for NLP

Overcoming feature engineering with representation learning

Raw input to machine learning (ML) models needs to be transformed to vectors in feature spaces that represent their most discriminative characteristics. In the field of NLP, the raw input typically consists of segments of linguistic utterances, e.g. characters, words, paragraphs, sentences, or documents. Typical feature sets in NLP include lexical features, syntactic features, and semantic features. The mapping from raw input to feature representations is the core of many ML problems and immense efforts have been spent to find suitable feature representations to best capture the input characteristics for the given task. As illustrated in the previous section, the room for possible features for QE is large due to the bilingual setting of the task. Designing this mapping appropriate for a certain task is commonly understood under the term "feature engineering". Once having defined a set of features for a particular task, the next challenge is to combine or weigh them in a most effective way, which typically involves techniques for feature selection or expansion. Furthermore, sparsity issues can occur, e.g. when a feature is not observed often enough to reliably contribute knowledge to the classifier. Often, large feature representations are mapped to lower-dimensional spaces (e.g. via Singular Value Decomposition (SVD) or Principal Component Analysis (PCA)) to facilitate efficient storage and computations. Depending on the task, the domain and the classifier, certain feature representations and techniques might be more suitable than others, so there is no set of globally best features and hence no globally best strategy to map the linguistic input to vectors. Deep learning presents an elegant solution to avoid manual feature engineering and instead learn a suitable mapping jointly with the ML task.
[Deng, 2014] compares several prevalent definitions of Deep Learning (DL), and extract two essential aspects: (1) DL is a machine learning technique for models with multiple layers of

non-linear information processing, which (2) implement learning of abstract feature representations at successively higher layers. Models for DL are designed with multiple ML layers ("deep architecture"), to capture factors of variation in the input at several layers of abstraction ("deep representation"). Amongst the wide array of deep learning architectures (see [Bengio, 2009] for a survey), we will exclusively consider feed-forward neural networks (NN) with multiple hidden layers and inspect how they can be applied to a NLP task like QE. How deep architectures can learn deep representations is addressed in Section 3.2.

The success of DL for NLP

The idea of replacing feature engineering by learning deep representations led to great improvements in various NLP tasks in the last decade. In the context of SMT, the most prominent example are language models, where neural network language models [Bengio *et al.*, 2013, Schwenk, 2007, Bengio *et al.*, 2006] consistently outperform the traditional approaches. Similar progress was made with translation models [Son *et al.*, 2012, Devlin *et al.*, 2014, Sundermeyer *et al.*, 2014]. For a comprehensive overview over the history of neural networks and deep learning we refer the reader to Schmidhuber's overview on DL in NN [Schmidhuber, 2015]. Since this work focuses on the application to QE, we only highlight some stations of the development of DL for NN that paved the way for the application to NLP tasks.

The origin of NN for unsupervised and supervised machine learning goes back to the development of artificial neural networks in the 1960s, where the motivation was to model signal processing in the brain (most prominently by [Rosenblatt, 1985] introducing the Perceptron as a model for a single neuron; the first multi-layer perceptron (MLP) was developed in 1965 by [Ivakhnenko & Lapa, 1966]. [Rumelhart *et al.*, 1986] presented a first solution for efficient training of NN via error backpropagation. In the late 1980s, mostly NN with few hidden layers were trained, and [Hornik *et al.*, 1989] found that a NN with a single layer of a sufficient number of units can approximate any multivariate continuous function with arbitrary accuracy. However, applications of deep NN did not yield extraordinary successes until the early 2000s. Problems with gradient descent training were identified: the vanishing or exploding gradients problem [Hochreiter, 1991, Hochreiter, 1998], and how to lead the learning towards good local optima for the non-convex optimization problem [Bengio *et al.*, 1994]. With new architectures like the Long Term Short Memory (LTSM) [Hochreiter & Schmidhuber, 1997] and the introduction of layer-wise unsupervised pre-training [Ranzato *et al.*, 2007] solutions to these problems were found and allowed more effective training of NN. Furthermore, faster machines and larger data volumes had a positive effect on the success

of DL techniques. An instance of the successful introduction of DL for various NLP word-level classification problems is examined in detail in Section 3.3.
The essential benefits of DL that recent development in NLP relies on can be summarized as follows:

- Techniques for DL allow to **learn representations** of the input data dispensing with manual time- and cost-consuming feature engineering.
- **Multi-layer** architectures represent the input information on several successive levels of abstraction.
- **Non-linear activation** of representation units allow a high complexity for representing relationships in the input data.
- DL benefits from massive amounts of unlabeled data when used for **unsupervised pre-training**.
- It offers effective methods for training using the **backpropagation** algorithm in supervised settings.

3.2 Learning Distributed Word Representations

Unsupervised word embeddings for supervised tasks

In the previous sections we explained that one of the main benefits of deep learning for NLP is that it offers a technique to avoid manual feature engineering. In the field of NLP, great attention was given to recent research on learning distributed word representations (also referred to as word embeddings).
The simplest way to represent a word as a vector is by one-hot encoding: All values in the vector are 0, except for one, which is at the index of the word within a finite vocabulary. The dimensionality of this vector hence equals the size of the vocabulary. More sophisticated approaches learn word representations by clustering or from co-occurrence matrices that captures statistics about the word occurring in different contexts [Turney & Pantel, 2010, Sahlgren, 2006]. Clustering approaches produce one-hot representations over a smaller vocabulary, whereas distributional approaches yield sparse vectors over a large number of contexts. These high-dimensional representations can be projected into a low-dimensional space via techniques for dimensionality reduction like SVD

or PCA, but these are computationally expensive and complicate incorporating new words into the representation vector space. Therefore, representation learning aims to directly train dense low-dimensional representations.
Since labeled data is sparse, and semi-supervised approaches require specific types of models and training regime [Turian *et al.*, 2010], unsupervised training of word representations from large amounts of unlabeled data is most attractive. If trained independently from any supervised task, they can be shared among researchers and application scenarios. [Turian *et al.*, 2010] demonstrate that various unsupervised word representations can improve state-of-the-art supervised systems for NLP tasks like chunking and named entity recognition when integrated as word features. Although the idea of globally useful word representations is appealing, they also found that different kinds of word representations are suitable for different tasks. [Weston *et al.*, 2008] take a different approach, where unsupervised word embeddings are trained jointly with a supervised task and improve the model's accuracy of the supervised task. Our QE from scratch approach builds on this finding and learns bilingual word embeddings as auxiliary task to the supervised task of QE.

Word embeddings trained with NN

Several strategies have been developed to use neural networks for training word embeddings. Typically, they constitute the basis for neural network language models (NNLM):

- [Bengio *et al.*, 2003] introduce a neural probabilistic language model that learns word embeddings on a NN architecture of two hidden layers: one shared word feature layer and an ordinary linear layer with a non-linear activation function. During training, the log-likelihood of the next word given previous words and their embeddings is maximized.
- [Collobert & Weston, 2008] present a neural language model that is very similar to the model by [Bengio *et al.*, 2003], but non-probabilistic. It is trained to discriminate n-grams present in the training corpus from corrupted n-grams while learning shared word embeddings. Their NN architecture is identical to the architecture that they use for training SENNA, a multi-purpose NLP-tagger [Collobert *et al.*, 2011] (see Section 3.3).
- [Mikolov *et al.*, 2013b] and [Mikolov *et al.*, 2013a] introduce two models, commonly known as `word2vec`: a continuous bag of words (CBOW) and a continuous skip-gram model. The NN architecture is similar to [Bengio *et al.*, 2003]'s feed forward NNLM, but misses the non-linear layer (for efficiency reasons). In the CBOW model, the current word is predicted

based on context, and in the skip-gram model the context words are predicted instead (see [Goldberg & Levy, 2014] for an explanation of the details). Both models are trained with a negative sampling technique that maximizes the likelihood of observed words and contexts in a training corpus and the likelihood that randomly sampled contexts are unobserved with the current word. [Levy & Goldberg, 2014] show that the skip-gram negative sampling model implicitly factorizes a word-context positive pointwise mutual information weighted matrix, shifted by some constant, which connects these models back to the count-based models by [Turney & Pantel, 2010].

Multilingual word embeddings

With parallel and comparable corpora available, the described approaches can easily be extended to learning representation for words in bi- or multilingual contexts. We follow the categorization scheme proposed by [Luong *et al.*, 2015] distinguishing between monolingual adaption, bilingual mapping, and bilingual training schemes.

- **Monolingual adaption**: [Zou *et al.*, 2013] learn bilingual embeddings with an objective like [Collobert & Weston, 2008] but extended by bilingual constraints. The idea is that starting from English embeddings, Chinese embeddings can be inferred from unsupervised alignment information. They use the trained embeddings to form a semantic similarity feature for phrase-based machine translation from Chinese to English.

- **Bilingual mapping**: [Mikolov *et al.*, 2013c] assume a linear relationship between languages and the corresponding word representations. They train a linear transformation matrix to map monolingual `word2vec` embeddings from one language to the other. The trained model yields high translation accuracy on English to Czech and vice-versa and English to Spanish. They suggest that this method can be used to correct dictionary errors in translations.
  [Klementiev *et al.*, 2012] train a probabilistic neural language model similar to [Bengio *et al.*, 2003] with a multitask learning approach: Words are considered as tasks and their relatedness inferred from alignment statistics is used as regularization term. The trained word embeddings are not shared across languages, although being jointly induced and aligned.
  In contrast to the above described approaches, [Hermann & Blunsom, 2014] do not rely on word alignments, but build their model on sentence-aligned parallel corpora. The idea

is that sentence-parallel texts share common information on the sentence level, not necessarily on the word level. Hence, representations of sentences and documents can be mapped cross-lingually as a composition of word embeddings. Their model is evaluated on two cross-lingual document classification (CLDC) tasks on eleven languages and outperforms the [Klementiev *et al.*, 2012] word embeddings on the CLDC taks for English and German. For monolingual document classification, they outperform the SENNA embeddings.

- **Bilingual training**: [Wolf *et al.*, 2014] refrain from the idea of a guaranteed linear transformation between embeddings of different languages. Their model jointly learns skip-gram embeddings for both languages without an explicit transformation but through word alignments.
  [Gouws *et al.*, 2014] enhance the model by [Klementiev *et al.*, 2012] by introducing a cross-lingual objective that is not based on prior computed word alignments, but only on the alignments of sentences in a parallel corpus.
  [Luong *et al.*, 2015] focus on training bilingual word embeddings in such a way that they can be useful for both monolingual and bilingual NLP tasks. They train a bilingual skipgram model (BiSkip) with negative sampling on a joint objective with monolingual and bilingual constraints. As extension to the monolingual skipgram model, BiSkip predicts neighbouring words cross-lingually given the alignment link between words of both languages. It achieves state-of-the-art performance on a German to English CLDC task, while still outperforming other bilingual embeddings (amongst them [Klementiev *et al.*, 2012], [Gouws *et al.*, 2014], [Hermann & Blunsom, 2014]) on monolingual word similarity tasks.
  [Gouws & Søgaard, 2015] rely neither on alignment information, nor on parallel data. Instead, their approach requires a mixed corpus (shuffled concatenation of source and target corpus) and a small number of bilingual task-specific equivalences, which can originate e.g. from word alignments and knowledge bases. Randomly substituting words in the mixed corpus with their equivalences, they train the standard CBOW model on this bilingual input. They demonstrate that a model trained on Wiktionary POS classes and WordNet equivalence classes performes well on cross-lingual POS tagging and supersense tagging tasks.

Evaluating and comparing word embeddings

From the large number of possible approaches to word embeddings the question arises how these embeddings can be compared or evaluated. Word representations are traditionally expected to reflect the semantics that the represented words carry, so that semantically similar words should have similar embeddings, i.e. their vectors should be close in the resulting embedding space. While it can be misleading to simply take a look at nearest neighbour words, present them in lists or visualize them in a two-dimensional space, quantitative means are needed to evaluate these embeddings more thoroughly. Therefore, a number of datasets for intrinsic semantic evaluation were created. These include word similarity tasks, where given a pair of words, their similarity is predicted. The predictions are compared to human ratings of similarity in terms of correlation metrics. Tasks for semantic similarity include e.g. the WordSim-353 [Finkelstein *et al.*, 2001] or the MC [Miller & Charles, 1991] dataset. Note that these datasets are mostly monolingual and if available in several languages, they have (to our knowledge) not been applied to measure cross-lingual similarity. Similar tasks are the word analogy tasks. These tasks require to predict suitable words for questions like "a is to b as c is to ?" or to score given analogy pairs. The analogies to find might be syntactically or semantically defined [Mikolov *et al.*, 2013d].
In addition to this intrinsic evaluation, there are various extrinsic evaluation scenarios. The idea behind these is to improve standard benchmarks on different NLP tasks by providing classifiers with new features derived from the trained word embeddings. For example, the CoNLL 2003 named entity recognition (NER) benchmark datasets is used evaluate NER classifiers that learn from a standard discrete feature set extended with continuous word features from trained word embeddings (e.g. in [Pennington *et al.*, 2014, Turian *et al.*, 2010]). By using fixed datasets, classifiers and discrete features, the results of the models enriched with different word embeddings allow a fair comparison of the effectivity of the word embeddings for these tasks.

Our QE model learns embeddings as a bilingual analog to [Collobert *et al.*, 2011], but benefits largely from `word2vec` embeddings trained on monolingual corpora. In comparison to the above described approaches, we make use of both word alignments (optional) and bilingual QE training data. The training of high-quality multilingual word embeddings as such is not the primary objective of our deep learning approach to QE, but we will include evaluations of the trained embeddings to gain insights into the learned embedding space in Section 5.3.

3.3 SENNA: NLP "Almost from Scratch"

As explained in previous sections, feature engineering is a tedious process, especially because there is no globally valid set of features across NLP tasks. [Collobert *et al.*, 2011] therefore motivate to "use a single learning system able to discover adequate internal representations." By this they aim to avoid task-specific feature engineering and instead find meaningful intermediate word representations that can be used for various NLP tasks. They develop the all-purpose tagger "SENNA" (Semantic/syntactic Extraction using a Neural Network Architecture) for NER, POS tagging, semantic role labeling (SRL) and chunking based on a neural network architecture that learns these tasks "almost" from scratch, i.e. with reduced reliance on prior NLP knowledge.

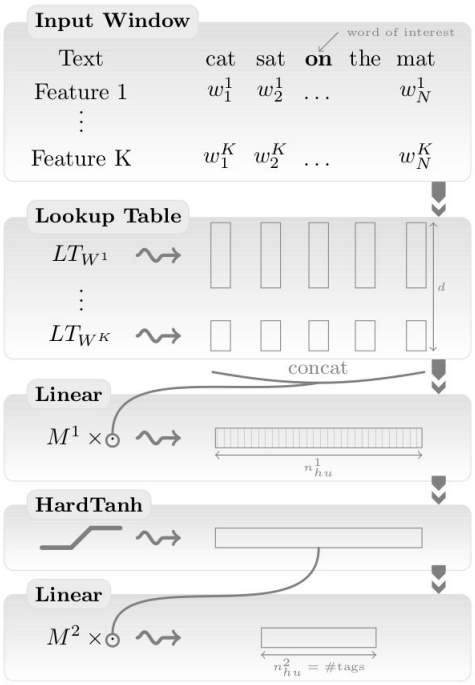

Figure 1: SENNA neural network architecture for the window approach. Source: [Collobert *et al.*, 2011]

Neural network architectures for NLP tagging on word level

In contrast to standard state-of-the-art NLP systems, [Collobert *et al.*, 2011] do not optimize their models for a single benchmark, but for multiple benchmarks, and they avoid task-specific

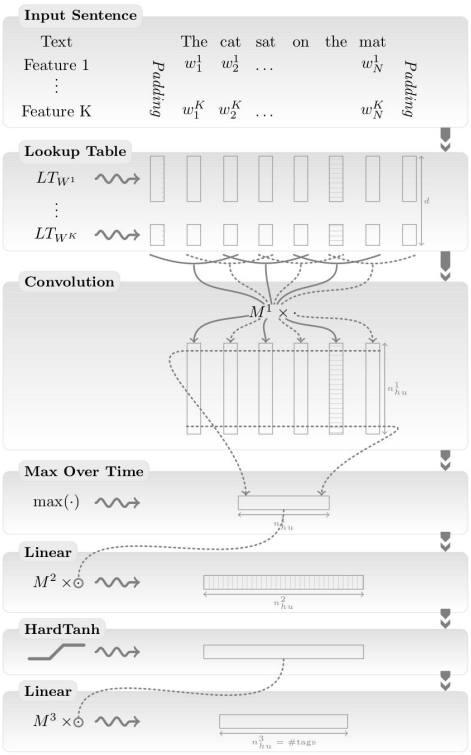

Figure 2: SENNA neural network architecture for the sentence approach. Source [Collobert *et al.*, 2011]

engineered features and dependencies on pre-existing systems. Also, they do not only build on manually annotated text corpora, but aim to leverage large amounts of unlabeled data. To this aim, they train NN models for four NLP tasks in end-to-end fashion with features that require minimal pre-processing.

Two suggested NN architectures are illustrated in Figures 1 and 2. In the window approach, the input layer extracts k features for each word within a window (centered around the word that is predicted). These features are further processed by a lookup-table layer that maps each feature to a d-dimensional feature vector. These feature vectors are concatenated and constitute the input to a standard linear layer. The output of this linear layer is a n_{hu}-dimensional vector. This vector is transformed with a $hardtanh$ activation function and subsequently fed into another linear layer. This layer produces a vector with one value for each possible tag. The values of this output vector can be interpreted as scores, when normalized with a softmax operation (Equation 10).

The window approach does not appear suitable for SRL, so they extend the architecture to consider the whole sentence when tagging a word. The architecture is roughly the same (compare Figures 1 and 2), but a convolutional layer is added between the lookup table layer and the first linear

layer. The convolution can be interpreted as a generalized version of the window approach: For each window extracted from the sentence, feature representations are looked up in the table and concatenated (local feature representation). They are subsequently fed into a linear layer with parameters shared across all windows. Since the size of the output of this layer depends on the number of windows and thus the length of the input sentence, a max operation over all output vectors (over "time" in terms of positions in the input sentence) is applied to obtain a global feature representation with a fixed dimensionality. The further processing works exactly as in the window approach.

Training the neural network

The training objective is to maximize the likelihood of the training data given the model. To compute this likelihood, [Collobert *et al.*, 2011] suggest two approaches. The word-level likelihood considers all words to be tagged independently. The output of the NN is a vector with one score s_t for each possible tag $t \in T$. Transforming these scores via a softmax operation [Bridle, 1990] over all tags allows to interpret the score as probability. The sentence-level likelihood on the other hand considers dependencies between successive tags of words in a sentence. Therefore, transition scores for jumping from one tag to another in successive words are added to the network scores to form a score on sentence-level. Although the sentence-level likelihood turns out to be superior in the experiments by [Collobert *et al.*, 2011], our approach described in 4.1 implements the word-level approach, so we will not further got into details for the alternative on sentence-level.
Training is realized with stochastic gradient ascent [Bottou, 1991, Rumelhart *et al.*, 1986] on iteratively selected random training samples $(x, y) \in D$ making gradient steps with a given learning rate λ:

$$\theta \leftarrow \theta + \lambda \frac{\partial \log p(y|x, \theta)}{\partial \theta} \tag{1}$$

The key to representation learning is that not only the parameters of the linear layers, but also the parameters of the lookup table layer are updated during training.
The neural network can be understood as composition of functions, where each layer l corresponds to one function f^l with parameters θ^l. The first layer is a function on the training input x, the second a function on the output of the first layer and so on. The input signal is passed through the network in a feed-forward manner:

$$f_\theta(x) = f_\theta^L(f_\theta^{L-1}(\dots f_\theta^1(x)\dots)) \tag{2}$$

where L is the number of layers in the NN. Given a loss function, e.g. the negative log-likelihood, its gradient can hence be recursively computed with respect to each θ^l using the chain rule for backpropagation [Rumelhart *et al.*, 1986]. Details of the backpropagation through the MLP and the computation of gradients for each layer follow in Section 4.2.

Experiments and Findings

With this training process, the deep NN models are trained on the four supervised CoNLL benchmark tasks. Input words are lowercased, occurrences of numbers are replaced by the string "NUMBER" and a feature for capitalization is introduced. The dictionary for the lookup table layer contains the 100,000 most frequent words of Wall Street Journal. Out-of-vocabulary (OOV) words are replaced by the string "RARE". The results on the benchmark tasks reveal that the NN models perform not as well as benchmark systems selected as baseline. The authors identify data sparsity in the training corpus (i.e. rare words) as one possible reason for inferior performance. To solve this problem they enhance the network by initializing the word lookup table with unsupervised word embeddings trained on large unlabeled corpora (Wikipedia and Reuters) with a NNLM as in [Collobert & Weston, 2008]. After the initialization, supervised training is performed as before. Note that the supervised training affects also the word embeddings, so that they are fine-tuned for the supervised tasks (cf. [Weston *et al.*, 2008]). This unsupervised pre-training of the word embeddings significantly improves the performance on the benchmark tasks, such that at least the POS baseline can be reached. Interestingly, the final word embeddings without pre-training appear to not make sense when judged by intuition (considering nearest neighbours in the embedding space), whereas the pre-trained final embeddings do. [Collobert *et al.*, 2011] further implement multitask learning for training the parameters for all four tasks jointly, which results in minimal improvements over the single-task systems. However, they cannot beat the baselines without adding additional features loosening their "from scratch" philosophy. As depicted in Figures 1 and 2 adding discrete features is realized with additional lookup tables, one for each feature type. The effect is that the NN learns not only embeddings for words, but also embeddings for the additional feature types, e.g. for POS tags.

3.4 Deep Learning for QE

As mentioned in Section 2.2 only a few approaches used neural networks for QE. [Shah *et al.*, 2015b] build a continuous space neural language model to extract features for sentence-level predictions. In the WMT15 QE evaluation these features alone yield slightly better results than the baseline features, and added to other features they consistently lead to improvements. Likewise, [Shah *et al.*, 2015a] perform a detailed analysis of the effectiveness of these types of features for sentence-level QE on both source and target side and come to similar results. In contrast, [de Souza *et al.*, 2014] use neural networks as classifiers for QE, not only for feature extraction. With their bidirectional long short-term memory recurrent neural network (BLSTM-RNN) they were ranked first in the WMT14 word-level evaluation. The recurrence in the network architecture allows to keep a history about the models states, and by bidirectional processing of the input the prediction for a word is dependent on the hidden states for all previous and all subsequent predictions of that sentence [Schuster & Paliwal, 1997]. The input to the BLSTM-RNN are features including word posterior, language model and word lexicon probabilities, POS tags and features based on confusion networks and on the QE labels of other granularities.

Like [de Souza *et al.*, 2014] we design a deep neural network classifier for word-level QE, but we only use raw words as input and train an unidirectional multi-layer perceptron without recurrence. This makes our approach similar to the continuous space neural language models, except for we have bilingual input (as suggested by [Shah *et al.*, 2015a]) and a discrete output value instead of a probability distribution over words. With our system combination we similarly to [Collobert *et al.*, 2011] address the question, where manually-designed features are superior to the latent feature representation of the deep neural network and what their individual contribution to a combined model is.

4 Quality Estimation from Scratch

We apply the framework of [Collobert *et al.*, 2011] to learn bilingual correspondences "from scratch", i.e. from raw input words. The following sections provide details about the NN architecture, the training process, and the design of system combination with a linear baseline feature classifier.

4.1 Neural Network Architecture

Our *QUality Estimation from scraTCH* (QUETCH) system is based on a neural network architecture built with `Theano` [Bergstra *et al.*, 2010][1]. We design a MLP architecture with one hidden layer, non-linear $\tanh$ activation functions and a lookup-table layer as proposed by [Collobert *et al.*, 2011]. The lookup-table has the function of mapping word to continuous vectors and is updated during training. Figure 3 illustrates the connections between the input, hidden lookup-table and linear layer, and the output. The softmax over the activation of these output units is interpreted as score for the two classes. Formal details about the layers, the parameters and the training are specified in the next two sections.

As briefly sketched in Section 3.3, feed-forward neural networks can generally be formalized as a nested composition of functions, where each function f_θ^l corresponds to a layer $l \in L$ of the network:

$$f_\theta(\cdot) = f_\theta^L(f_\theta^{L-1}(\dots f_\theta^1(\cdot)\dots)) \tag{3}$$

The input for the network is the input to the innermost function f_θ^1 representing the first layer, whose output is input to the next function representing the next layer, etc. The output of the outmost function f_θ^L representing the last layer, is the output of the network. This mechanism allows feeding forward the input signal through all the successive layers of the network. The signal is transformed at each layer, and if non-linearity is involved in these transformations, the output signal is not linearly reconstructable from the input.

1 Code available at `https://github.com/juliakreutzer/quetch`

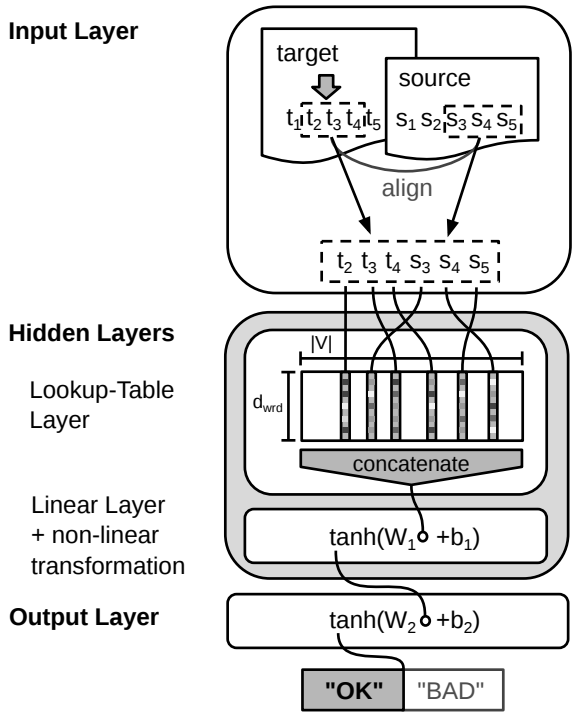

Figure 3: QUETCH. Source: [Kreutzer *et al.*, 2015]

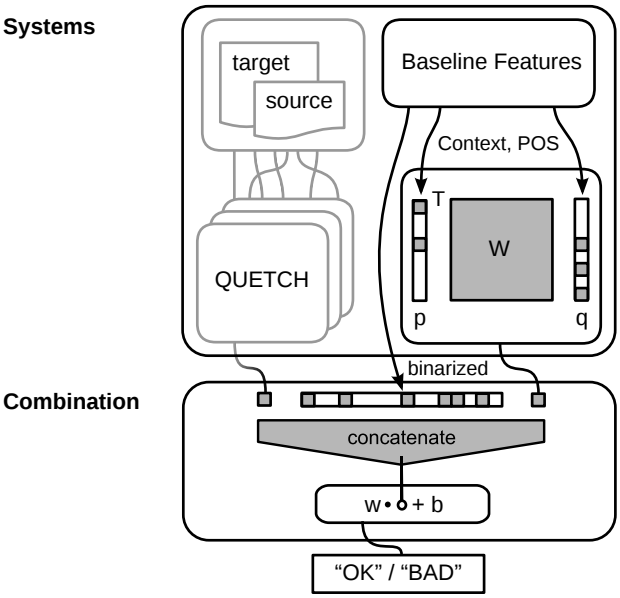

Figure 4: QUETCH+. Source: [Kreutzer *et al.*, 2015]

The input to our network consists of:
(1) a tokenized and lowercased[2] target sentence $t = [t_1 t_2 \dots t_T]$ of length T,
(2) a tokenized and lowercased source sentence $s = [s_1 s_2 \dots s_S]$ of length S,
(3) an indicator $i \in [0, T]$ to the position of the target word t_i (focus word) that has to be labeled, and
(4) an alignment function $a : i \rightarrow j$ that maps the target position i to a source position $j \in [0, S]$.
Note that this alignment function does not have to be a linguistically sound alignment function as used in SMT – it only requires to be fully defined on the target side. In the following, $x = \langle t, s, i, a \rangle$ denotes the input to the network.

From words to vectors All target and source words are indexed within a vocabulary V and represented as one-hot encoded vector of length V. From the input sequences t and s context words are extracted around the positions i and j. For this purpose we define a context extractor function $c_{win_{src},win_{tgt}}(\cdot)$ which is parametrized by the source window size win_{src} and the target window size win_{tgt}[3]

$$c_{win_{src},win_{tgt}}(x) = \begin{bmatrix} t_{i-\lfloor win_{tgt}/2 \rfloor} \\ \vdots \\ t_i \\ \vdots \\ t_{i+\lfloor win_{tgt}/2 \rfloor} \\ s_{a(i)-\lfloor win_{tgt}/2 \rfloor} \\ \vdots \\ s_{a(i)} \\ \vdots \\ s_{a(i)+\lfloor win_{tgt}/2 \rfloor} \end{bmatrix} \tag{4}$$

The result of the application of the context extractor function on an input is a $(win_{tgt}+win_{src})\times|V|$ sparse context matrix C of one-hot row vectors, each row representing one context word.
For example, consider a single sentence pair with $s =$ "The cat sat on the bike" and $t =$ "Die Katze saß auf dem Fahrrad". The vocabulary V contains all word types (lowercased) of the two sentences: $V =$ {the, cat, sat, on, bike, die, katze, saß, auf, dem, fahrrad}.

2 Apart from tokenization and lowercasing, no additional pre-processing was applied to the raw input data.
3 We assume that window sizes are uneven integers larger than 1 such that they can be centered at the target position i or the source position j respectively. If the window exceeds the sentence, a special PADDING token is inserted.

They are represented as one-hot-encodings:

$$\begin{aligned}
\text{the} &: [100000000000], \\
\text{cat} &: [010000000000], \\
\text{sat} &: [001000000000], \\
\text{on} &: [000100000000], \\
\text{bike} &: [000010000000], \\
\text{die} &: [000001000000], \\
\text{katze} &: [000000100000], \\
\text{saß} &: [000000010000], \\
\text{auf} &: [000000001000], \\
\text{dem} &: [000000000100], \\
\text{fahrrad} &: [000000000010], \\
\text{PADDING} &: [000000000001]
\end{aligned} \tag{5}$$

Assuming that a is the identity function, context window sizes of 3 yield the following context matrix for the target word $t_1 =$ "Katze":

$$c_{3,3}(t, s, 1, a) = \begin{bmatrix} 000001000000 \\ 000000100000 \\ 000000010000 \\ 100000000000 \\ 010000000000 \\ 001000000000 \end{bmatrix}$$

Word embeddings One-hot representations of words are sparse and not informative, since all words representations have equal distances in the embedding space. We aim to find dense representations that capture bilingual distributional knowledge such that words similar in meaning have similar representations. These word embeddings are stored in a lookup-table (LT) matrix $M^{|V| \times d_{wrd}}$ and are updated during training for QE after random initialization or unsupervised pre-training.

$$f_{LT}(x) = vec_{row}(Mx)^T \tag{6}$$

The lookup table operation (Equation 4.1) is realized by a vectorized dot product of context matrix and embedding matrix and returns a $d_{wrd} * (win_{tgt} + win_{src})$-dimensional lookup vector, where vec_{row} represents the row-major vectorization of the input matrix: $vec_{row}(A) = a_{11} \dots a_{1m} \dots a_{21} \dots a_{2m} \dots a_{n1} \dots a_{nm}$. Intuitively, this operation concatenates slices of the lookup table representations for all context words, as illustrated in Figure 3.

Standard affine layers The lookup vector is input to the first affine layer

$$f_{lin}(x) = b_1 + W_1\, x \tag{7}$$

which is parametrized by a bias vector b_1 of length hu and a $hu \times |d_{wrd} * (win_{tgt} + win_{src})|$ weight matrix W_1. We speak of the hyper-parameter hu as the number of hidden units of this layer. To introduce non-linearity into the neural network, we apply the hyperbolic tangens $tanh$ as non-linear activation function elementwise to the output of the linear layer. The $tanh$ returns a value in the range $[-1, 1]$.

$$tanh(x_i) = \frac{\exp^{2x_i - 1}}{\exp^{2x_i + 1}} \tag{8}$$

Another linear layer

$$f_{out}(x) = b_2 + W_2\, x \tag{9}$$

parametrized by a $k \times hu$-dimensional matrix W_2 forms the output layer of the whole neural network architecture. The number of units of this layer equals the number of output labels k, i.e. $k = 2$ in the case of a binary prediction ($Y = \{\mathsf{OK}, \mathsf{BAD}\}$).

From activations to scores To make the outputs of the neural network interpretable as label probabilities, we apply an elementwise softmax normalization function to the k-dimensional output vector of the second affine transformation:

$$score(x_i) = \frac{\exp^{x_i}}{\sum_{j=1}^{k} \exp^{x_j}} \tag{10}$$

For each label we hence obtain a probability score.

Feed forward A full feed-forward pass through all the layers of the network is a nesting of the above defined functions:

$$N_\theta(x) = f_{out}(tanh(f_{lin}(f_{LT}(c_{win_{src}win_{tgt}}(x))))) \tag{11}$$

$$= b_2 + W_2\, tanh(b_1 + W_1\, (vec_{row}(M \cdot c_{win_{src},win_{tgt}}(x)))) \tag{12}$$

The predicted label $\hat{y}$ for a given input x is the $arg\max$ of the network output:

$$\hat{y} = arg\max_{y_i \in Y} score(N_\theta(x)_{y_i}) \tag{13}$$

4.2 Training

Maximum likelihood objective Trainable parameters of the NN architecture are the bias vectors (b_1,b_2) and weight matrices (W_1,W_2) of the linear layers and the matrix $M \in R^{d_{wrd} \times |V|}$ that represents the lookup-table. When not explicitly referred to, they are denoted by θ. Let θ^l denote the parameters of a single layer l. Tunable hyper-parameters are the number of units of the hidden linear layer, the lookup-table dimensionality d_{wrd} and the learning rate.
The log-likelihood given the training data D consisting of labeled training samples (x, y) is defined as follows:

$$L_\theta = \sum_{(x,y) \in D} \log p(y|x, \theta) \tag{14}$$

The training objective is to maximize this log-likelihood, that is to find $\hat{\theta} = arg\max_\theta L_\theta$. The likelihood is defined as the softmax model score for the true label ([Collobert *et al.*, 2011]'s word-level likelihood):

$$p(y|x, \theta) = score(N_\theta(x)_y) \tag{15}$$

Using the definition of the softmax normalization (Equation 10) the log-likelihood can be rewritten in terms of the NN feedforward output $N(x)$.

$$L_\theta = \sum_{(x,y)\in D} \log p(y|x,\theta) \tag{16}$$

$$= \sum_{(x,y)\in D} \log score(N_\theta(x)_y) \tag{17}$$

$$= \sum_{(x,y)\in D} \log \frac{\exp^{N_\theta(x)_y}}{\sum_{j=1}^{k} \exp^{N_\theta(x)_j}} \tag{18}$$

$$= \sum_{(x,y)\in D} N_\theta(x)_y - \log \sum_{j=1}^{k} \exp^{N_\theta(x)_j} \tag{19}$$

SGA For maximizing (15) we use stochastic gradient ascent. With each training sample (x, y) at timestep t, the parameters are updated into the direction of the gradient of the objective function with stepsize λ:

$$\theta_{t+1} = \theta_t + \lambda \frac{\partial L_\theta}{\partial \theta_t} \tag{20}$$

$$= \theta_t + \lambda \frac{\partial \log score(N_\theta(x)_y)}{\partial \theta_t} \tag{21}$$

$$= \theta_t + \lambda \frac{\partial (N_{\theta_t}(x)_y - \log \sum_{j=1}^{k} \exp^{N_{\theta_t}(x)_j})}{\partial \theta_t} \tag{22}$$

Modular computation of gradients In order to compute the gradients for the parameters of each layer, the chain rule is applied repeatedly. Parts of the chain are the same for inner partial derivatives (the so-called backpropagated error signal), which allows for a modular definition of the derivatives. For each layer[4] it is hence sufficient to compute the gradient w.r.t. (1) the layer's parameters, and (2) the layer's input. Given a nested functions that defines the neural network (see Equation 3), gradients for all layers l can be computed from the gradient with respect to this layer's input $\frac{\partial L}{\partial f_\theta^l}$ and the gradient with respect to own parameters $\frac{\partial f_\theta^l}{\partial \theta_l}$:

$$\frac{\partial L}{\partial \theta^l} = \frac{\partial L}{\partial f_\theta^l} \frac{\partial f_\theta^l}{\partial \theta_l} \tag{23}$$

4 Here, "layers" are functions, so also the non-linear activation function is considered as a layer.

Depending on the depth of the layer, the gradient w.r.t its inputs might require applying the chain rule again:

$$\frac{\partial L}{\partial f_\theta^l} = \frac{\partial L}{\partial f_\theta^{l+1}} \frac{\partial f_\theta^{l+1}}{\partial f_\theta^l} \tag{24}$$

The detailed gradients for all layers are given in Section A.1.

4.3 Unsupervised Pre-training

Usually, the parameters of a neural network are initialized with zeros or random numbers, i.e. no a-priori knowledge is captured in the network. However, the learning process can benefit from knowledge that is encoded into the architecture prior to training [Saxe *et al.*, 2011] by being guided to good local minima in non-convex optimization [Erhan *et al.*, 2010, Bengio, 2009]. In case of QE, the model profits from seeing well-written source and target sentences – before actually seeing translations. `word2vec` [Mikolov *et al.*, 2013a, Mikolov *et al.*, 2013b] offers efficient methods to pre-train word representations in an unsupervised fashion such that they reflect word similarities and relations. Initializing the lookup-table with pre-trained `word2vec` vectors allows us to incorporate prior linguistic knowledge about source and target language into QUETCH. During the learning process, these representations are further optimized for QE and the vocabulary encountered during training. Note that other submissions to the WMT15 QE task also utilize features based on `word2vec` ([Shah *et al.*, 2015b]), but in a static way, such that the word embeddings are not optimized during training for QE.

4.4 Baseline Features and System Combination

The WMT15 data contains a number of baseline features (listed in Table 10). In order to exploit this additional information, a straightforward approach would be to integrate the baseline features in the deep learning system on the same level as word-features and train lookup-tables for each feature class [Collobert *et al.*, 2011]. While this certainly works for cetagorical features like POS tags, this is not suitable for continuous numerical features. Preliminary tests of extending QUETCH with a lookup-table for POS-tags did not result in better F1 scores. Also, training takes considerably longer, because of (1) the additional lookup-table to train and (2) the larger dimensionality of the vector representing a target word with its context. For these reasons, we decided to design a system combination that treats the QUETCH system and the baseline features

individually and independently. To this aim, we train separate systems, one based on the DL approach described in Section 4.1 (QUETCH), and one based solely on the baseline features. In a final step, the outputs of both systems are combined with binarized versions of selected baseline features. This combination is the input to a linear classifier that learns weights for these features (QUETCH+). On the one hand, system combination is a generally effective strategy for complex applications (see e.g. for MT [Heafield & Lavie, 2011, Karakos *et al.*, 2008] or for cross-lingual information retrieval [Schamoni & Riezler, 2015]). On the other hand, the modular, linear combination provides additional knowledge about the individual contributions. This will help to address the questions how much feature-engineering is necessary for QE, which features are most useful, and whether the information represented by QUETCH is already present in some of the baseline features.

Baseline Features System

To obtain a system for baseline features that is most complementary to QUETCH, we use the Vowpal Wabbit (VW) toolkit [Goel *et al.*, 2008] to train a linear classifier, i.e. a single-layer perceptron. We build new features by "pairing" baseline features, thus we quadratically expand the original feature space and learn a weight for each possible pair.

Assuming two feature vectors $\mathbf{p} \in \{0,1\}^P$ and $\mathbf{q} \in \{0,1\}^Q$ of sizes P and Q where the n^{th} dimension indicates the occurrence of the n^{th} feature, we define our linear model as

$$f(\mathbf{p}, \mathbf{q}) = \mathbf{p}^\top W \mathbf{q} = \sum_{i=1}^{P} \sum_{j=1}^{Q} p_i W_{ij} q_j. \tag{25}$$

The weight matrix $W \in R^{P \times Q}$ encodes relations between features [Bai *et al.*, 2010, Schamoni *et al.*, 2014]. The value of $f(\cdot, \cdot)$ is the prediction of the classifier given a target vector $\mathbf{p}$ and a vector of related features $\mathbf{q}$.

To address the problem of data sparsity, we reduced the number of possible feature pairs by restricting the feature expansion to two groups: (1) *target words* are combined with *target context words* and *source aligned words*, and (2) *target POS tags* are combined with *source aligned POS tags*.

System Combination

For the final system combination, we train another linear classifier. The combined systems comprises 82 features: the QUETCH-score, the score predicted by the baseline feature system, and the remaining 80 features are binary features derived from the baseline feature set. The QUETCH-score is the system's prediction combined with its likelihood, for the baseline feature model we directly utilize the raw predictions with clipping at ± 1. Binarized features were inserted to enrich the classifier with additional non-linearity. They consist of (1) the binary features from the baseline feature set, and (2) binned versions of the numerical features from the same set. For small groups of discrete values we assigned a binary feature to each possible value, for larger groups and real-valued features we heuristically defined intervals ("bins") containing roughly the same number of instances. The integration of the single components for the system combination is illustrated in Figure 4.

5 Experiments

5.1 WMT QE Tasks

In this section, we describe the training of the QUETCH model on WMT data and report results of experiments that led to the WMT15 submission. With experiments on both WMT14 and WMT15 data we are able to analyse the performance of our model on four language pairs, two different kind of gold labels, two different data set sizes, and in a system combination with baseline features.

As described in section 2.1 the WMT word-level QE tasks consist of predicting the word-level quality level of machine translations, without the use of human references, and without insight into the machine translation system. We focus on binary labels for quality ("OK"/"BAD"), since these are available for both WMT14 and WMT15 data. Main evaluation metric for both evaluations is the F1-score of BAD instances. Tables 3 and 4 present the definition for true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for the classification of OK and BAD instances respectively. From these, Precision (P), Recall (R), and the F1-score ($F1$) are computed as defined in Equation 26.

$$
\begin{aligned}
P &= \frac{TP}{TP + FP} \\
R &= \frac{TP}{TP + FN} \\
F1 &= 2\,\frac{P \cdot R}{P + R}
\end{aligned}
\tag{26}
$$

		Prediction	
		OK	BAD
True	OK	TP	FN
	BAD	FP	TN

Table 3: Confusion matrix for OK-classification.

		Prediction	
		OK	BAD
True	OK	TN	FP
	BAD	FN	TP

Table 4: Confusion matrix for BAD-classification.

Sentence id	Token id	Token	Multi-level	L1	Binary
3.1	0	This	OK	OK	OK
3.1	1	group	OK	OK	OK
3.1	2	of	OK	OK	OK
3.1	3	galaxies	OK	OK	OK
3.1	4	called	OK	OK	OK
3.1	5	Arp	OK	OK	OK
3.1	6	273	OK	OK	OK
3.1	7	,	Punctuation	Fluency	BAD
3.1	8	took	Mistranslation	Accuracy	BAD
3.1	9	up	Mistranslation	Accuracy	BAD
3.1	10	the	OK	OK	OK
3.1	11	space	Word_order	Fluency	BAD
3.1	12	telescope	Word_order	Fluency	BAD
3.1	13	Hubble	OK	OK	OK
3.1	14	for	OK	OK	OK
3.1	15	Nasa	OK	OK	OK
3.1	16	.	OK	OK	OK

Table 5: Example of WMT14 de-en target sentence annotation.

WMT14

The WMT14 data covers translations of several thousand training instances from German to English (de-en), Spanish to English (es-en), and vice versa (en-de, en-es). The following example from the de-en training data (sentence with id 3.1) illustrates the characteristics and the format of the data. For the source sentence "Diese Gruppe von Galaxien, Arp 273 genannt, nahm das Weltraumteleskop Hubble für die Nasa auf.", the target annotations are listed in Table 5. All tokens of the target sentences were annotated with MQM categories, that were used to infer L1 and binary labels. In this example word order, mistranslation and punctuation errors cause 5 of 17 tokens being labeled BAD. Note that punctuation and word order errors are errors that could be detected by a standard target-side language model, whereas the mistranslation error cannot be identified without knowing the source text. Learning both from local target and source context, the QUETCH model should in principle be able to cover both types of errors.

Table 6 presents an overview over the distribution of labels in training and test data across the four language pairs. The skewness of the data varies from language to language. The WMT evaluation focuses on the classification quality for the BAD class, but the model is trained with a

	Language Pair	Total Words	% OK	% BAD
Training Set	de-en	6,453	79.06	20.94
	en-de	11,281	70.48	29.52
	es-en	19,912	83.43	16.57
	en-es	47,411	64.87	35.13
Test Set	de-en	1,777	78.61	22.39
	en-de	2,368	71.33	28.67
	es-en	3,114	82.37	17.63
	en-es	9,613	64.38	35.62

Table 6: Distribution of binary labels on WMT14 datasets for binary word-level QE.

maximum likelihood objective that treats both classes equally. This is why the skewness heavily influences the results when measured with BAD F1. A solution for this effect will be addressed in the experiments for WMT15.

Since the plain QUETCH system does not rely on language-specific features, we simply use the same deep learning architecture for all of these language pairs.
QUETCH is trained on the WMT14 training set, with a source and target window size of 3, a lookup-table dimensionality of 10, 300 hidden units, and a constant learning rate of 0.001[1]. Test and training data were lowercased and tokenized[2] and aligned with the `fast_align` tool from the `cdec` toolkit [Dyer *et al.*, 2010]. The collection of corpora provided with WMT13's translation task[3] served as source for unsupervised pre-training of the word embeddings: Europarl v7 [Koehn, 2005], Common Crawl corpus, and News Commentary. The initial word embeddings were trained jointly for all language pairs in the hope that this will lead to embeddings that suit multilingual tasks best. Therefore we concatenated the above listed corpora and use `gensim`'s [Řehůřek & Sojka, 2010] `word2vec` CBOW implementation for training on all words of the concatenated corpus with a context window of five.

Following the WMT14 evaluation [Bojar *et al.*, 2014], we report on accuracy and BAD F1-score, the latter being the task's primary evaluation metric. The WMT14 baselines trivially predict either only BAD or only OK labels. Table 7 presents the best F1-scores during training and the according accuracies for QUETCH under different configurations.

1 This hyper-parameter set was chosen by grid search on the WMT15 development data. The primary goal was to build a competitive system for the WMT15 evaluation, so the WMT14 data served mainly the purpose of exploring other language pairs and translations, and hyper-parameters were not tuned for WMT14 specifically.
2 Training and test target data was delivered tokenized, source data was not tokenized.
3 `http://www.statmt.org/wmt13/translation-task.html`

	Configuration	**BAD F1**	**Accuracy**
en-es	(v)	0.4378	0.5087
	(a)	0.4164	0.5107
	(p)	0.5206	0.4026
	(a), (p)	**0.5228**	0.4196
es-en	(v)	0.2197	0.7604
	(a)	0.2470	0.7749
	(p)	0.3203	0.8051
	(a), (p)	**0.3396**	0.8076
en-de	(v)	0.3743	0.6090
	(a)	0.4197	0.6381
	(p)	0.4684	0.6060
	(a), (p)	**0.4863**	0.6271
de-en	(v)	0.2482	0.7001
	(a)	0.2426	0.6837
	(p)	0.3734	0.6657
	(a), (p)	**0.3791**	0.6792

Table 7: QUETCH results on WMT14 task 2 test data under different configurations: (v)anilla system, (p)retraining of word embeddings, (a)lignments from an SMT system.

The plain QUETCH system yields an acceptable accuracy, but the BAD F1-scores are not competitive. Adding alignment information further improves the accuracy for all language pairs but de-en. It improves the F1-score only for es-en and en-de, which indicates that the model is still prone to local optima. It is in fact pre-training that boosts the BAD F1-score – this initial positioning in the parameter space appears to have a larger impact on the training outcome than the introduction of translation knowledge via alignments. However, we can achieve further improvement when combining both pre-training and alignments. As a result, QUETCH outperforms the official winning systems of the WMT14 QE task (see Table 9) and the trivial baselines for all language pairs. The fact that the overall tendencies are consistent across languages demonstrates that QUETCH is capable of language-independent quality estimation.

In addition to the above experiments, we train a multilingual QUETCH model on the concatenation of all WMT14 datasets jointly and test it on the concatenated test sets ("all") and the individual language pairs. The results in Table 8 indicate that this approach is not superior to the independent training for each language pair in terms of BAD F1-score for the individual language pairs.

	Model	**BAD F1**	**Acc.**
all	(a)(p)	0.4288	0.6396
en-es	(a)(p)	0.3079	0.5614
es-en	(a)(p)	0.2320	0.6429
en-de	(a)(p)	0.3179	0.5832
de-en	(a)(p)	0.2698	0.6528

Table 8: Multilingual learning for all WMT14 language pairs. The model is trained on a multilingual corpus that contains the WMT14 training data for all language pairs.

	Submission	**BAD**	**Acc.**
en-es	FBK-UPV-UEDIN/RNN	0.4873	0.6162
es-en	RTM-DCU/RTU-GLMd	0.2914	0.8298
en-de	RTM-DCU/RTU-GLM	0.4530	0.7297
de-en	RTM-DCU/RTU-GLM	0.2613	0.7614

Table 9: Winning submissions of the WMT14 Quality Estimation Task 2 [Bojar *et al.*, 2014].

WMT15

For the WMT15 QE word-level task, training, development and test data with annotations for errors as edit operations (replacements, insertions, or deletions) with respect to human post-edits [Snover *et al.*, 2006] were provided. Table 11 presents two example sentences of the WMT15 training data and illustrates how the QE labels are inferred from comparing the target and the postedition. Note that the binary label scheme based on edits is very strict and agnostic about the severity of the error in the translation, e.g. "humana" is labeled "BAD" although only the gender is incorrect ("humano").

In addition to the labeled data, a set of 25 baseline features that operate on source and target translation, but do not use features of the SMT pipeline that produced the translations, was supplied. For an overview of the baseline features see Table 10. Note that the the extraction of these features requires additional resources like WordNet for polysemy counts, language models in both source and target language, alignments, and pre-processing by a POS tagger, a named entity recognizer, and a machine translation system for generating the pseudo-references. These dependencies on external resources and processors can have an effect on the quality of the resulting QE model, but this has not been evaluated yet. However, the features allow us to evaluate the approach for system combination introduced in Section 4.4.

The distribution of binary labels is strongly skewed towards “OK” labels as listed in Table 12,

Category	Feature
Token Counts	source token count
	target token count
	source target token count ratio
Context	token
	left context
	right context
Alignment	first aligned token
	left alignment
	right alignment
Token Class	is stopword
	is punctuation
	is proper noun
	is digit
Language Model	highest order ngram left
	highest order ngram right
	backoff behavior left
	backoff behavior middle
	backoff behavior right
	source highest order ngram left
	source highest order ngram right
MT	pseudo-reference
POS-tag	target pos
	aligned source pos list
Polysemy	polysemy count source
	polysemy count target

Table 10: Baseline features provided with the WMT15 word-level QE task data.

even more so than in the previous QE task at WMT14 (increased by a factor of approximately four).

For the experiments on the WMT15 en-es data, we introduce a weight w for BAD training samples, such that QUETCH is trained on each BAD sample w times. In this way, we easily counterbalance the skewed distribution of labels, without modifying the classifier's loss function. Also, we utilize the larger and non-parallel raw Wikicorpus [Reese *et al.*, 2010] in English and Spanish for pre-training (around 600 mio words for English, around 120 mio words for Spanish, dump of 2006).

During training of the VW-system, we experiment with various loss functions (hinge, squared, logistic) and find the model trained on squared loss to return the highest accuracy. Unwanted collisions in VW's hashed weight vector were reduced by increasing the size of the hash to 28 bits. To prevent the model from degenerating towards OK-labels, we utilize VW's option to set the weight for each training instance individually and increase the weights of the BAD-labeled

source	Roasted pumpkin , pumpkin seeds , pumpkin pie , you name it .
target	Calabaza tostadas , semillas de calabaza , pastel de calabaza , lo que sea .
postedition	Calabaza rostizada , semillas de calabaza , tarta de calabaza , dilo tu .
labels	OK BAD OK OK OK OK OK BAD OK OK OK BAD BAD BAD OK
source	Both are too passive to really ascribe human-like behavior .
target	Ambos son demasiado pasivos con el comportamiento realmente atribuyen similar a la humana .
postedition	Ambos son demasiado pasivos para asignarle comportamiento humano .
labels	OK OK OK OK BAD BAD OK BAD BAD BAD BAD BAD BAD OK

Table 11: Two examples from the WMT15 training data. Tokens labeled "BAD" are underlined.

	Total Words	**% OK**	**% BAD**
Training Set	257,548	80.85	19.15
Development Set	23,207	80.82	19.18
Test Set	40,899	81.12	18.88

Table 12: Distribution of binary labels on data set for WMT15 word-level QE

instances to 4.

The VW-system and the system combination are trained in a 10-fold manner, i.e. the VW-system is trained on 9 folds and the weights for system combination is tuned on the 10^{th} fold of the training data. The final weights of the model for evaluation are averaged over all 10 folds.

Table 13 presents the results on the WMT15 data for both QUETCH, the baseline feature VW model, and the system combination referred to as QUETCH+. The QUETCH results were produced under the same parameter conditions as in the WMT14 experiments, and the newly introduced *w* is set to 2 for the submitted and the combined model, and to 5 for another model that was explicitly designed for a high BAD F1-score.

Although proceeding in the same manner as in the WMT14 experiments, we see slightly different tendencies here: Adding alignments has a positive effect on the BAD F1-score, whereas pre-training improves mainly the accuracy. Still, the combination of both yields both a high BAD F1-score and a high accuracy, which indicates that QUETCH succeeds in integrating both contributions in a complementary way. Adding BAD weights furthermore improves the BAD F1-score, yet losing some accuracy. Further increasing the weight up to 5 strengthens this effect, such that we obtain a model with very high BAD F1-score, but rather low accuracy.
The stand-alone VW model yields generally higher BAD F1-score, but does not reach QUETCH's accuracy. To enhance the orthogonality of the two models for combination, we select a QUETCH

	Configuration	**BAD F1**	**Accuracy**
QUETCH	(v)	0.2535	0.7104
	(a)	0.2628	0.7099
	(p)	0.2535	0.7668
	(a), (p)	0.2793	0.7716
	†(a), (p), (w=2)	0.3527	0.7508
	(a), (p), (w=5)	0.3876	0.6031
	‡(a), (p), (w=2)	0.2985	0.7888
	‡VW	0.4084	0.7335
	†QUETCH+	**0.4305**	0.6977

Table 13: QUETCH results on en-es WMT15 task 2 test data under different configuration setting: (v)anilla model vs. models using (p)re-training, (a)lignments from an SMT-System, and (w)eighting of the BAD-instances. Submitted systems are preceded by †, components of the final QUETCH+ system are marked with ‡.

model with high accuracy for the system combination. Interestingly, the system combination appears to profit from both models, resulting in the overall best BAD F1-score. The resulting VW weights of 1.188 for QUETCH and 0.951 for VW underline each system's contribution. The next most important features for the combination were *pseudo_reference* and *is_proper_noun* with weights of 0.2208 and 0.1557, respectively.

System	**BAD F1**	**OK F1**	**All F1**
baseline	0.1678	0.8893	0.7531
QUETCH	0.3527	0.8456	0.7526
QUETCH+	0.4305	0.7942	0.7256
UAlacant/OnLine-SBI-Baseline	0.4312	0.7807	0.7147

Table 14: Official test results on WMT15 task 2 for word-level translation quality. The All F1-score is the weighted average of BAD F1 and OK F1, where the weights are determined by the frequency of the classes in the test data. The UAlacant/OnLine-SBI-Baseline and the QUETCH+ predictions show no significant difference at p=0.05 and are both announced official winners.

Table 14 shows the final evaluation results on the WMT15 task 2 for the main evaluation metric of F1 for predicting BAD word-level translation quality, the F1 for predicting OK translations and their weighted average. Both submitted systems, QUETCH and QUETCH+, yield considerable improvements over the baseline. The QUETCH+ system that combines the neural network with the linearly weighted baseline features is nominally outperformed by one other system by

0.07% BAD F1 points, but their difference is not significant at p=0.05, tested with approximate randomisation [Padó, 2006].

Model Selection

We observe that the training process after the first iterations produces high BAD F1-score models, then further improves the accuracy whilst slowly decreasing the BAD F1-score. This is due to the fact that we do not optimize on the BAD F1-score directly, but the log-likelihood of the data, which is skewed towards the OK label. This behavior allows us to select models with individual trade-offs between BAD F1-score and accuracy at different stages of training. Figures 5 and 6 illustrate this process: They depict the accuracy and the BAD F1-score for early stopping within the first 200 epochs of training for the vanilla, the aligned, the pre-trained and two weighted models. The accuracy for the un-weighted models start high with around 80% and then slowly decreases, probably an effect of overfitting. The good starting values are due to the high ratio of "OK" instances in the data. The higher the weight for the "BAD" instances, the lower the accuracy and the higher the BAD F1-score starts. Whereas all models converge to a consistent accuracy of around 75% after 200 epochs, the models initialized with pre-trained word embeddings show a clearly higher BAD F1 score. Naturally, one would select a model at a point where the evaluation results are stable and close to convergence, but since we need a model with primarly high BAD F1-score, we exploit the early peak in the BAD F1 curve for weighted models for the submission to the WMT evaluation (early stopping at epoch 15). For the system combination, we selected the model after 38 epochs.

5.2 Additional Experiments

A small number of additional experiments with the QUETCH architecture were conducted to give directions for further development. In contrast to the results reported in the previous sections, the results reported in this section are not based on exhaustive meta-parameter tuning and do not aim to compete with state-of-the-art models. The purpose of these experiments is to explore both further possible applications and limitations of the model.

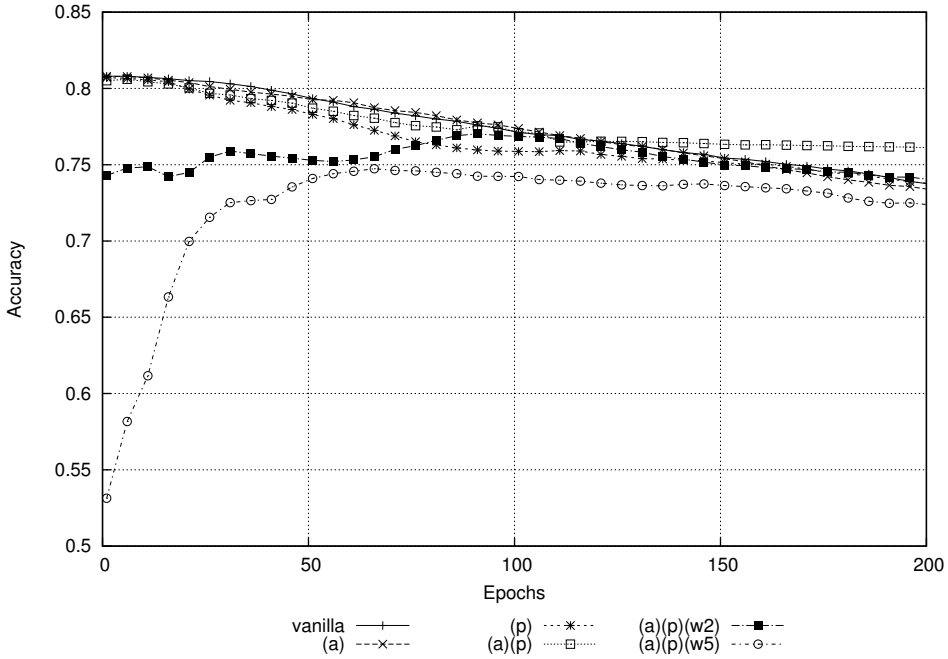

Figure 5: Development of Accuracy during training, evaluated on dev data

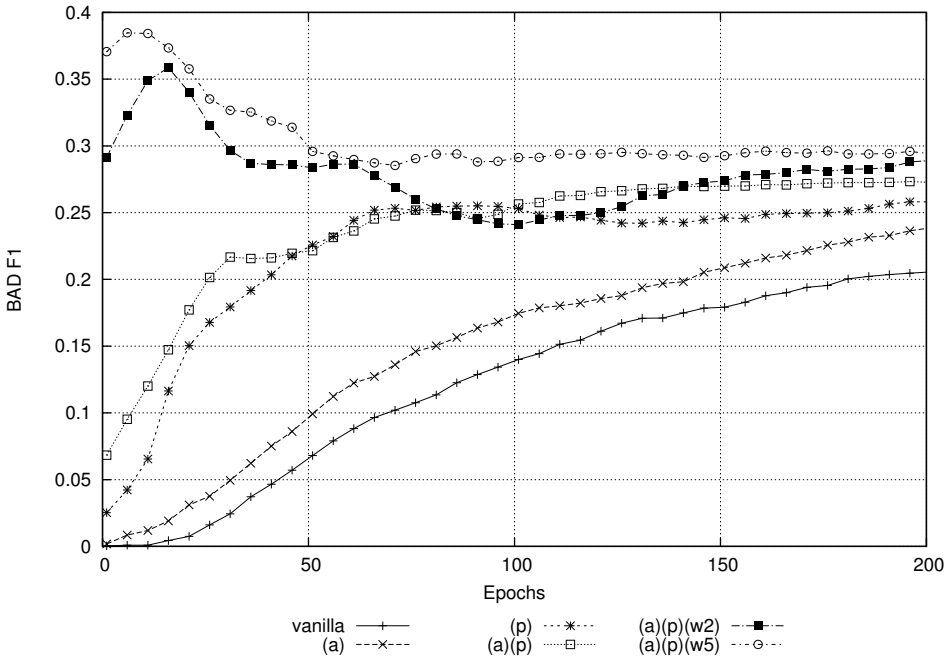

Figure 6: Development of BAD F1 during training, evaluated on dev data

Language Pair	Configuration	**MAE**	**RMSE**
en-es	(p)	0.4983	0.7517
es-en	(p)	0.66	0.9487
en-de	(p)	0.64	0.9237
de-en	(p)	0.6733	0.8981

Table 15: QUETCH results on WMT14 task 1.1. test set

Transfer to Sentence-Level Predictions

The architecture for word-level QE predictions can easily be adapted to sentence-level predictions. On the sentence level, the input to the NN consists of full source and target sentences and the output is a single score, depending on the task formulation. We limit these first experiments to the formulation of the WMT14 task 1.1, where sentence scores are three categories describing levels of post-editing effort:[4]

- 1: perfect translation, no post-editing needed at all
- 2: near miss translation: translation contains at most 2-3 errors, and possibly additional errors that can be easily fixed (capitalisation, punctuation)
- 3: very low quality translation, cannot be easily fixed

Transferring from word-level to sentence-level predictions, the NN architecture remains unchanged. Instead of feeding in the representations of context windows, we now feed in the representation of whole sentences. Technically, we simply choose the size of the context windows in such a way that they cover the whole sentences, i.e. at least the longest sentence in the training data. Padding is added to shorter sentences. The only change in the output is that we have one additional quality label, so we increase the size of the output layer by one.[5] For the experiments on the WMT14 task 1.1 data we set context sizes to 60 and the number of hidden units to 100. We found that a smaller number of hidden units shows clear benefits here. Since the input to the network is much larger than in the word-level setting, a higher compression for the successive layer appears to be beneficial for the generalization performance. Table 15 reports mean absolute error (MAE) and root mean squared error (RMSE) for the model's predictions.

4 See `http://www.statmt.org/wmt14/quality-estimation-task.html` for the detailed task description.

5 If the sentence-level task is not understood as a classification, but as a regression task (as for example in the WMT15 task 1, where HTER scores are predicted), the output layer can alternatively be reduced to one output unit without softmax normalization.

Language Pair	Configuration	**MAE**	**RMSE**
en-es	Baseline	0.5200	0.6600
en-es	Winning System	0.4900	0.6100
es-en	Baseline	0.5700	0.6800
es-en	Winning System	0.5300	0.6400
en-de	Baseline	0.6400	0.7600
en-de	Winning System	0.5800	0.6800
de-en	Baseline	0.6500	0.7700
de-en	Winning System	0.5500	0.6700

Table 16: Baseline and winning systems on WMT14 task 1.1. test set

The approach to sentence-level QE via the above described changes to the word-level approach is naive and the results (compare Table 15 and Table 16) emphasize the need for more sophisticated NN architectures that are specifically designed for the sentence-level task. Problematic in our approach are the large proportion of padding vectors that are added to the input representation for short sentences, and furthermore the fact that all words in both sentences contribute equally to the sentence pair representation. These issues illustrate that NN architectures that allow variable-length input sequences and weighted contributions of input elements could offer attractive solutions for QE (e.g. bidirectional recurrent NN [Sundermeyer *et al.*, 2014], or the encoder-decoder model with attention mechanism [Bahdanau *et al.*, 2015] that were successful for SMT).

Learning Parameters

After the WMT15 submission, more experiments with a wider range of meta-parameters for the learning process were conducted. We found that the use of the `sigmoid` activation function (27) instead of `tanh` improves the classification results. We hypothesize that the `sigmoid`'s range of [0,1] suits the latter transformation into probabilities better than the `tanh`'s range of [-1,1]. Table 17 presents a comparison of models with `sigmoid` and rectifier (ReLU) activation functions (28) evaluated on the WMT15 development data. For comparison, the results for a linear model are also reported. Furthermore we found considerable improvement when shuffling the training data before each epoch. Regularizing the l2-norm of the parameters however did not result in better classification quality (not reported here). Still these results indicate that a further tuning of the learning process and NN properties offer room for improvement over the QUETCH models submitted to the official WMT evaluation.

Configuration	**BAD**	**Accuracy**
(a)(p)(w=2) + identity	0.3022	0.7710
(a)(p)(w=2) + `ReLU`	0.3258	0.7696
(a)(p)(w=2) + `tanh`	0.3527	0.7508
(a)(p)(w=2) + `tanh` + shuffle	**0.3760**	0.7259
(a)(p)(w=2) + `sigmoid`	0.3644	0.7622
(a)(p)(w=2) + `sigmoid` + shuffle	0.3729	0.6606

Table 17: QUETCH results on WMT15 task 2 test data with different activation functions and optional shuffling before each epoch. The results for shuffling are averaged across 3 independent runs. The "identity" activation function makes the model linear.

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{27}$$

$$ReLU(x) = max(0, x) = \begin{cases} 0 & x < 0 \\ x & \text{otherwise} \end{cases} \tag{28}$$

5.3 Evaluation of the Word Embeddings

So far we found evidence that the QUETCH model successfully learns to make predictions for QE, but we cannot directly inspect the knowledge that it gained, since the values of the hidden layer's matrices and vectors are not linguistically interpretable. However, an inspection of the learnt word embeddings will allow us to at least get an idea of how the model represents the meaning of single words in the context of QE. With the following experiments we address the questions (1) how words are represented and (2) whether these representations capture a general cross-lingual knowledge that could be useful for other cross-lingual tasks.

As in Section 5.2 the goal of the results reported in this section is not to compete with other systems, because the model was not designed and optimized for these tasks. By inspecting the parameters of the lookup table layer in isolation, we only look at the first step of transforming the input words into abstract representations for QE. It would be a completely different experiment to transform the QUETCH architecture to be trained for the evaluation tasks below directly. The experiments of this section are all based on the models trained for WMT15 on translations from English to Spanish.

The QE Vector Space

Our first analysis aims to find out how words are located in the bilingual vector space that the QE lookup table represents. To this aim we list nearest neighbours for selected words and visualize the similarity between words via projections on a two-dimensional space with truncated SVD.
Nearest neighbours are inspected for four selected words: one from English ("love"), one from Spanish ("amor"), and two that are common in both languages ("internet", "drama"). The representations of these words are compared to all other word representations in the vector space: the word representations with the highest cosine similarity are considered nearest neighbours. Comparing the `word2vec` embeddings and the bilingual vanilla and submitted QUETCH embeddings, we make the following observations: The `word2vec` embeddings reflect some conceptual and contextual similarities (e.g. "internet"-"online", "love"-"my") of the English training corpus, but not on the Spanish side (see Table 18). Also, the lists for the words common in both languages only contain English words. From that we learn that the way we trained the `word2vec` embeddings (concatenating English and Spanish corpora, details see Section 4.3) might not have been optimal, since `word2vec` embeddings are usually good in reflecting those similarities (compare e.g. [Mikolov *et al.*, 2013a]). Initializing the QUETCH embeddings with these embeddings, we obtain

embeddings with similar characteristics – the the model is fine-tuned on the QE training data, the more changes the ranking of the nearest neighbours and the more Spanish neighbours are found. However, when initializing the embeddings randomly, the similarities appear to stay random (or unintuitive), although the model performs well on QE. That shows that the embeddings used for initialization of the QE embeddings do not necessarily have to represent word similarities in an intuitive way for yielding good QE results. The similarities reflected in the QUETCH vector space have their origin in the `word2vec` initialization, not from QE training.
What happens during QE training is that the initial `word2vec` embeddings that clearly separate English and Spanish embeddings as illustrated in Figure 7 are moved closer together (compare Figures 8 and 9). During `word2vec` training, English and Spanish words do not (or only rarely) share contexts, whereas during QE training, they are observed and updated jointly. This can be understood as adding cross-lingual knowledge to the `word2vec` embeddings. The question remains whether these updates are appropriate for `word2vec` embeddings, because QUETCH and `word2vec` architectures and training contexts differ, which will be addressed in the next two sections.

Word Similarity Tasks

As illustrated in the previous section, the `word2vec` embeddings do not reflect any cross-lingual knowledge. However, when fine-tuned with QUETCH, they are updated in cross-lingual context. Inspecting the top ten nearest neighbours of a few selected words, we did not find any cross-lingual similarities in the resulting QUETCH embeddings. Evaluating the embeddings quantitatively on standard word similarity tasks will show whether cross-lingual similarities are induced by QUETCH fine-tuning of `word2vec` embeddings. Therefore we compare the performance of the embeddings on the wordsim353 [Finkelstein *et al.*, 2001] and the MC [Miller & Charles, 1991] word similarity tasks. Both tasks involve predicting the similarity of a given pair of words. The predictions are compared to human annotations by measuring the Pearson correlation between annotations and predictions. Both datasets originally contain only English words, but translated versions for Spanish are available [Hassan & Mihalcea, 2009]. From these translations we create a cross-lingual word similarity task: given a word in one language and a word in another language, predict their similarity. For example, the translated pairs “book” - “library” (en), “libro” - “biblioteca” (es) form the cross-lingual pairs “libro” - “library” (es-en) and “book” - “biblioteca” (en-es). Note that the performance of `word2vec` and QUETCH embeddings cannot be compared directly, as QUETCH embeddings are trained on a larger dataset and also use alignment information. We therefore also

include word2vec models trained only on WMT15 data and on both Wikicorpus and WMT15 data to measure the influence of the training corpus genre and size. Similarity between words is measured by cosine similarity between their vector representations. The QUETCH embeddings are the ones of the submitted model. The results in Tables 19 and 20 may indicate that during QE training cross-lingual similarities are learned, but some monolingual similarities are lost.[6] Furthermore, we again find evidence that the Spanish word2vec embeddings do not reflect similarities as well as the English ones. The QUETCH embeddings based on these embeddings perform surprisingly well on the monolingual Spanish similarity tasks.

Cross-Lingual Document Classification

After having evaluated the QUETCH embeddings against human judgement of word similarity, we now continue with an extrinsic evaluation of the QE models in another down-stream task, namely cross-lingual document classification (CLDC). We therefore conduct experiments similar to [Hermann & Blunsom, 2014] on the TED corpus. The TED corpus contains English transcriptions of TED conference talks and their translations[7]. All documents are labelled with multiple keywords, so predicting these keywords originally is a multi-class multi-label task, but here it is reduced to a multi-class classification task (see [Hermann & Blunsom, 2014]). With this corpus, document classification can be evaluated both monolingually (predict the keyword of a document after having trained on documents of the same language) and cross-lingually (predict the keyword of a document after having trained on documents of a different language). The corpus contains 1038 documents for training and 99 for testing. [Hermann & Blunsom, 2014] reduce the keywords per document to the 15 most frequent keywords, which are: technology, culture, science, global, design, business, entertainment, arts, politics, education, art, health, creativity, economics, biology. Instead of training one multi-class classifier for all classes, we train 15 One-vs-Rest classifiers, one for each keyword. The features of each document are composed of the word embeddings. Like [Huang *et al.*, 2012] we use the average of the word vectors weighted by the words' inverse document frequency (idf). All our models are linear SVMs with a hinge loss objective and are built with `Scikit-learn` [Pedregosa *et al.*, 2011] and are trained for 10 epochs with SGD. Additionally we introduce class weights to balance the distribution of keywords. The models reported in [Hermann & Blunsom, 2014] build on word embeddings with a dimensionality of 128. Therefore we train another QUETCH model with the same parameter settings as the one

6 We assume that the large variations for the performance on the MC tasks is due to the small size of the dataset.
7 Available at `http://www.clg.ox.ac.uk/tedcldc/`

submitted to WMT15, but with larger word embeddings ($d_{wrd} = 128$). Classification performance is measured with macro-averaged F1 scores over all classes. The results (see Table 21) on the monolingual document classification task once more support the impression that the trained embeddings are not suitable for monolingual tasks. Again we find that QE training improves the quality of the Spanish `word2vec` embeddings. In the cross-lingual setting (see Table 22), the SVM models based on QUETCH embeddings actually show a very strong performance, in particular in the translation direction English to Spanish, which was also the direction for QE training. However, the `word2vec` embeddings outperform the QUETCH embeddings in the opposite direction.

In conclusion, these experiments have shown that the knowledge gained during QE training – manifested in the word embeddings – can be beneficial for other cross-lingual tasks, in particular for tasks with the same translation direction. Monolingual similarities that are reflected in the initial `word2vec` embeddings are not preserved during QE training. We also found that our `word2vec` embeddings might not be optimally trained (especially for Spanish), so exchanging them for other embeddings that better suit the cross-lingual setting might improve the QUETCH model.

Word	Vanilla	word2vec	QUETCH "submitted"	QUETCH "late"
love	tiró	fool	blowjob	great-looking
	hotspots	dreaming	asshole	assholes
	marsupial	boobs	sheepish	reads
	frightened	my	stupid	dreaming
	slice	blowjob	queasy	idiots
	gazelle-like	asshole	ahem	goddamn
	posibilidad	dreams	snuggles	iou
	gah	sheepish	everybody	bitch
	pre-cambio	yesterday	boobs	hearted
	capitalist	stranger	cuddle	lovin
amor	óptica	camisa	sabe	falló
	pin	narración	partir	similar
	settlement	triste	existe	ye
	stories	existe	triste	twitter
	infieles	suerte	querer	figuratively
	bali	mano	aquí	blandos
	acquihire	alguien	vengo	fuiste
	airports	mujer	dolor	proles
	cebo	sabe	miedo	jot
	balderton	hombre	vino	something
internet	dilapidated	database	microsoft	2gb
	disfuncional	digital	irc	anchors
	comprobaciones	tripadvisor	digital	aol
	decomissioned	google	desktop	obvi
	twins	omg	sharepoint	weekday
	soon	gumtree	linux	commercials
	reciprocal	online	all-in-one	mornings
	tropos	info	mozilla	updated
	chace	salesforce.com	wordpress	itunes
	relativo	virtual	software	billboards
drama	guard	film	brit	soccer
	filipinas	comedy	film	sundance
	layer	cabaret	comedy	program
	cosmonaut	music	classics	music
	especializa	brit	threepenny	penguin
	thankful	thriller	cabaret	hbo
	sensational	threepenny	documentary	version
	luck	documentary	thriller	orchestra
	madgesty	classics	penguin	soap-opera
	giving	soap-opera	soap-opera	incitación

Table 18: Example of nearest neighbors for 10-dimensional `word2vec` and QUETCH embeddings after training for QE. Similarity is measured by the cosine between the vectors. The vanilla model was trained for 323 epochs, the submitted QUETCH model is evaluated at epoch 15 and 345 epoch (QUETCH "late").

		en		es	
Embeddings		WordSim353	MC	WordSim353	MC
QUETCH	(a)(p)(w)	0.1189	0.2462	**0.4489**	**0.1303**
word2vec	Wiki	**0.3289**	**0.4739**	-0.1584	-0.0003
	WMT15	0.2154	0.4214	-0.0754	0.0643
	Wiki+WMT15	0.095	0.0238	-0.0759	0.0404

Table 19: QUETCH and word2vec embeddings evaluated on English and Spanish monolingual word similarity tasks

		en-es		es-en	
Embeddings		WordSim353	MC	WordSim353	MC
QUETCH	(a)(p)(w)	**0.1647**	**0.2331**	**0.2194**	0.0409
word2vec	Wiki	0.1501	0.079	0.1758	-0.1751
	WMT15	0.0934	-0.1603	0.1614	**0.4078**
	Wiki+WMT15	-0.027	0.1379	0.1131	0.0808

Table 20: QUETCH word2vec embeddings evaluated on cross-lingual word similarity tasks from English to Spanish and vice versa

Embeddings	en-en	es-es
Raw Data NB	0.481	0.526
Senna	0.4	
Polyglot	0.382	0.418
BiCVM	0.475 (DOC/ADD joint)	0.472 (DOC/ADD joint)
word2vec	0.1659	0.1712
QUETCH (a)(p)(w)	0.1628	0.2311

Table 21: F1-scores for the TED monolingual document classification task with training on QUETCH lookup table embeddings, compared to results by [Hermann & Blunsom, 2014] (BiCVM)

Embeddings	en-es	es-en
BiCVM	0.451 (DOC/BI joint)	0.446 (ADD single)
MT System	0.518	0.486
word2vec	0.4658	0.4350
QUETCH (a)(p)(w)	0.5479	0.4011

Table 22: F1-scores for TED cross-lingual document classification task with training on QUETCH lookup table embeddings, compared to results by [Hermann & Blunsom, 2014]

6 Discussion

This section presents a discussion about (1) the task of QE as presented in WMT, (2) the proposed approach and further improvements that could be addressed in future work.

QE at WMT As briefly outlined in Section 2.1, the task of QE is a relatively young task in the history of WMT. Several human evaluation scenarios have been tried to get ground truth labels for QE. This introduced inconsistencies in how the datasets were created: there were direct manual annotations in WMT14, whereas labels for QE were derived from human post-editions indirectly for WMT15. This reflects two different perspectives on translation quality, either an independent absolute judgment or judgment relative to a post-edition. However, one of the goals of QE is to predict quality without comparison to a reference, so deriving the labels from these references (posteditions) can be understood as a contradiction against this principle. Furthermore, one can argue that the comparison to a single postedition per sentence is not sufficiently reliable (like for SMT).

The system evaluation and ranking methods of WMT QE evaluations on sentence-level have been subject to criticism in [Graham, 2015]. This work suggests replacing RMSE and MAE for system ranking by the Pearson correlation metric, for it is unit-free and invariant to changes in scale and location. In the WMT15 findings [Bojar *et al.*, 2015] introduce a new metric for word-level QE system evaluation which is designed to overcome the shortcomings of the two-class F1 scores and to give credit to correct sequences of predictions. As we demonstrated, the BAD-F1 score can be pushed artificially whilst losing Accuracy and OK-F1, which is not prohibited by the task description but might not reflect the original purpose of the task.

Further improvement of the models The from-scratch approach presented in this thesis is effective, but fairly simple. In the following aspects it could get more complex and sophisticated to improve classification quality:

- The words are labelled independently. Taking previous predictions into account for subsequent predictions can be beneficial if e.g. whole phrases are translated incorrectly, as motivated in [Bojar *et al.*, 2015]. To overcome this independency assumption, we suggest adopting

the sentence-level log-likelihood scoring approach by [Collobert *et al.*, 2011] or using a BLSTM-RNN model similar to [de Souza *et al.*, 2014] with raw input. According to the new metric for word-level QE evaluation introduced by [Bojar *et al.*, 2015], our QUETCH submission outperforms the QUETCH+ submission, but is only ranked second. Systems that treat word-level QE prediction as a sequence labelling task, where consecutive labels are considered interdependent, are clearly preferred by this metric.

- The model only considers local context on both target and source side, so there is the risk that relevant context for the target word lies outside this context. For our experiments, relatively small contexts yielded the best results (with or without alignments), but this could be because of the similarity of Spanish and English word order.
- The learning rate is held constant during training. Preliminary experiments with decreasing learning rates did not improve the results, but adaptive per-feature learning rates like AdaGrad [Duchi *et al.*, 2011] or AdaDelta [Zeiler, 2012] might offer further improvements.
- In the QUETCH model the baseline features are ignored, but we showed that at least some features really add knowledge to the model with the system combination. The QUETCH+ model requires separate training and tuning of both QUETCH and baseline features system, a careful selection of a suiting QUETCH model, and tuning of the combined model, which can be tedious and prohibits a end-to-end training process. To overcome these issues, the features could directly get integrated into the neural network architecture like in the models by [Collobert *et al.*, 2011] or [de Souza *et al.*, 2014]. The challenge here is to find a suitable way for feature representation. The raw input features are integers and can be understood as indices to a lookup table, whereas some of the baseline features are real-valued numbers, which either need to be mapped to integers to allow for lookup table representations, or can be directly appended to the real-valued vector which is the result of the lookup opteration. Another option for lexical features would be to directly attach them to the raw input and jointly represent them in a lookup table, e.g. POS tags and language identifiers ("love_NN_en", "love_VV_en", "internet_NN_es"). However, this would probably lead to sparsity issues and prohibits learning feature and word representations independently.
- The method for pre-training the word embeddings was not tuned to best fit the QE training. The weak performance of the embeddings for the Spanish word similarity task indicates that the setting in which we trained them was not optimal. Training English and Spanish embeddings separately would probably already improve the quality of the Spanish

embeddings. Also, there are a number of parameters for `word2vec` training that can be adapted to improve the embeddings. Another idea is to pre-train them with the QUETCH architecture directly with a ranking objective to distinguish between observed and randomly sampled contexts (bilingual analogue to [Collobert *et al.*, 2011]). Alternatively, one could use bilingual word embeddings like those described in Section 3.2 for the initialization.

7 Conclusion

This thesis presents a novel neural network model for word-level quality estimation that was proven successful in the WMT15 QE evaluation. Having outlined the recent development of deep learning for NLP in general and QE in particular, we presented our motivation for building a deep neural network model for QE that learns continuous space representations for bilingual word contexts. In contrast to previous approaches, this model is as such independent of additional linguistic resources or pseudo-references by learning QE from raw bilingual input. Our QUETCH model presents an elegant solution to improve supervised QE training on a small dataset by unsupervised pre-training on large monolingual resources. With the experiments on WMT data and the successful participation in the WMT15 evaluation campaign, we proved that the QUETCH model is capable of competitive word-level QE across datasets and languages. A system combination of the QUETCH model and a strong linear classifier built on baseline features outperformed both individual models and illustrated the orthogonality of these models. Evaluating the QUETCH word representations on another two cross-lingual tasks, we found that these bilingual representations capture knowledge than can be useful for tasks beyond QE. Although the achieved results are promising, we see this work as a initial proposal of learning QE from scratch and hope for future improvements with the directions pointed out in the discussion.

A Appendix

A.1 Gradients

Let f^{in} denote the input to each layer and θ_l the layer's parameters. Note that the outputs of all layers are computed during feed forward pass (see Paragraph 4.1) and are necessary for the computation of the gradients with the equations below.

1. **Output layer** f_{out}

   With respect to parameters:

$$\frac{\partial L}{\partial \theta^{f_{out}}} \tag{29}$$

   That is:

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial f_{out}} \frac{\partial f_{out}}{\partial W_2} \tag{30}$$

$$= \frac{\partial L}{\partial f_{out}} f_{out}^{in} \tag{31}$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial f_{out}} \frac{\partial f_{out}}{\partial b_2} \tag{32}$$

$$= \frac{\partial L}{\partial f_{out}} \tag{33}$$

$$\tag{34}$$

   With respect to input f_{out}^{in} which is the output of the softmax layer (10):

$$\frac{\partial L}{\partial f_{out}^{in}} = \frac{\partial L}{\partial f_{out}} \frac{\partial f_{out}}{\partial f_{out}^{in}} \tag{35}$$

$$= \frac{\partial L}{\partial f_{out}} W_2 \tag{36}$$

2. **Hidden linear layer** f_{lin}

   With respect to parameters:

$$\frac{\partial L}{\partial \theta^{f_{lin}}} \tag{37}$$

   That is:

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial f_{lin}} \frac{\partial f_{lin}}{\partial W_1} \tag{38}$$

$$= \frac{\partial L}{\partial f_{lin}} f_{lin}^{in} \tag{39}$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial f_{lin}} \frac{\partial f_{lin}}{\partial b_1} \tag{40}$$

$$= \frac{\partial L}{\partial f_{lin}} \tag{41}$$

   With respect to input f_{lin}^{in} which is the output of lookup table operation f_{LT}:

$$\frac{\partial L}{\partial f_{lin}^{in}} = \frac{\partial L}{\partial f_{lin}} \frac{\partial f_{lin}}{\partial f_{lin}^{in}} \tag{42}$$

$$= \frac{\partial L}{\partial f_{lin}} W_1 \tag{43}$$

3. **Tanh layer**

   W.r.t input $tanh^{in}$ which is the output of the hidden layer f_{lin} (7). Note that this gradient is computed elementwise, because the $tanh$ function is applied elementwise.

$$\frac{\partial L}{\partial [f_{tanh}^{in}]_i} = \frac{\partial L}{\partial tanh} \frac{\partial tanh}{\partial [f_{tanh}^{in}]_i} \tag{44}$$

$$= \frac{\partial L}{\partial tanh} (1 - tanh^2([f_{tanh}^{in}]_i)) \tag{45}$$

4. **Word-level likelihood**

   W.r.t input L^{in} which is the output of the hidden layer f_{out} (9). Note that this gradient is

computed elementwise, because the underlying softmax normalization operates elementwise.

$$\frac{\partial L}{\partial [f_L^{in}]_i} = \frac{\partial (N_\theta(x)_y - \log \sum_{j=1}^{k} \exp^{N_\theta(x)_j})}{\partial N_\theta(x)_i} \tag{46}$$

$$= 1_{i=y} - \frac{\partial \log \sum_{j=1}^{k} \exp^{N_\theta(x)_j}}{\partial N_\theta(x)_i} \tag{47}$$

$$= 1_{i=y} - \frac{\partial \log \sum_{j=1}^{k} \exp^{N_\theta(x)_j}}{\partial \sum_{j=1}^{k} \exp^{N_\theta(x)_j}} \frac{\partial \sum_{j=1}^{k} \exp^{N_\theta(x)_j}}{\partial N_\theta(x)_i} \tag{48}$$

$$= 1_{i=y} - \frac{1}{\sum_{j=1}^{k} \exp^{N_\theta(x)_j}} \frac{\partial \sum_{j=1}^{k} \exp^{N_\theta(x)_j}}{\partial N_\theta(x)_i} \tag{49}$$

$$= 1_{i=y} - \frac{1}{\sum_{j=1}^{k} \exp^{N_\theta(x)_j}} \frac{\exp^{N_\theta(x)_i}}{1} \tag{50}$$

$$= 1_{i=y} - \frac{\exp^{N_\theta(x)_i}}{\sum_{j=1}^{k} \exp^{N_\theta(x)_j}} \tag{51}$$

$$= 1_{i=y} - score(N_\theta(x)_i) \tag{52}$$

5. **Lookup Table Layer** f_{LT}

   W.r.t. parameters M:

$$\frac{\partial L}{\partial \theta^{f_{LT}}} \tag{53}$$

   That is:

$$\frac{\partial L}{\partial M} = \frac{\partial L}{\partial f_{LT}} \frac{\partial f_{LT}}{\partial M} \tag{54}$$

$$= \frac{\partial L}{\partial f_{LT}} vec_{row}(C) \tag{55}$$

A.2 Word Embedding Visualizations

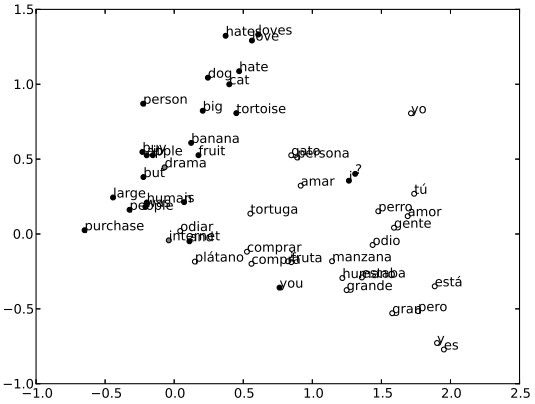

Figure 7: Word representations learned by the `word2vec` model that was used to initialized the late and the submitted model, projected into 2-dimensional space with truncated SVD

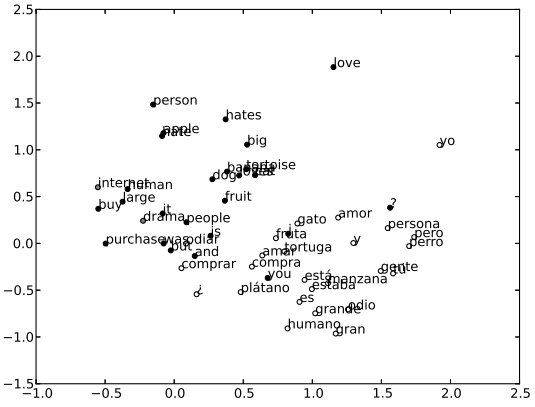

Figure 8: Word representations learned by the (a)(p)(w) late model after 345 epochs, projected into 2-dimensional space with truncated SVD

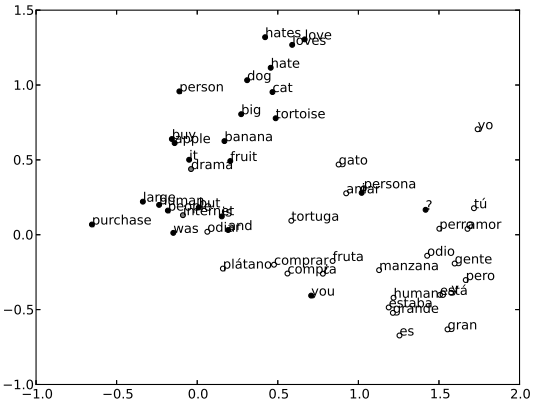

Figure 9: Word representations learned by the (a)(p)(w) submitted model after 15 epochs, projected into 2-dimensional space with truncated SVD

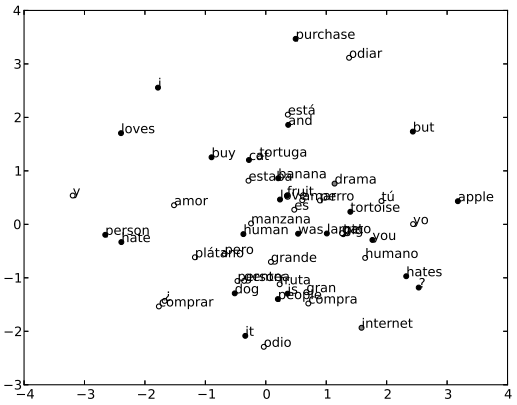

Figure 10: Word representations learned by the vanilla model after 323 epochs, projected into 2-dimensional space with truncated SVD

Bibliography

[Bach *et al.*, 2011] Bach, Nguyen, Huang, Fei, & Al-Onaizan, Yaser. 2011. Goodness: A method for measuring machine translation confidence. *Pages 211–219 of: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics.

[Bahdanau *et al.*, 2015] Bahdanau, Dzmitry, Cho, Kyunghyun, & Bengio, Yoshua. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. *In: Proceedings of the International Conference on Learning Representations (ICLR)*.

[Bai *et al.*, 2010] Bai, Bing, Weston, Jason, Grangier, David, Collobert, Ronan, Sadamasa, Kunihiko, Qi, Yanjun, Chapelle, Olivier, & Weinberger, Kilian. 2010. Learning to Rank with (a Lot of) Word Features. *Information Retrieval Journal*, **13**(3), 291–314.

[Banerjee & Lavie, 2005] Banerjee, Satanjeev, & Lavie, Alon. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. *Pages 65–72 of: Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, vol. 29.

[Bengio, 2009] Bengio, Yoshua. 2009. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, **2**(1), 1–127.

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

[Bengio *et al.*, 2003] Bengio, Yoshua, Ducharme, Réjean, Vincent, Pascal, & Janvin, Christian. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, **3**, 1137–1155.

[Bengio *et al.*, 2006] Bengio, Yoshua, Schwenk, Holger, Senécal, Jean-Sébastien, Morin, Fréderic, & Gauvain, Jean-Luc. 2006. Neural probabilistic language models. *Pages 137–186 of: Innovations in Machine Learning*. Springer.

[Bengio *et al.*, 2013] Bengio, Yoshua, Courville, Aaron, & Vincent, Pierre. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**(8), 1798–1828.

[Bergstra *et al.*, 2010] Bergstra, James, Breuleux, Olivier, Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, & Bengio, Yoshua. 2010. Theano: a CPU and GPU Math Expression Compiler. *In: Proceedings of the Python for Scientific Computing Conference (SciPy)*.

[Biçici & Way, 2014] Biçici, Ergun, & Way, Andy. 2014. Referential Translation Machines for Predicting Translation Quality. *Pages 313–321 of: Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland USA: Association for Computational Linguistics.

[Bojar *et al.*, 2014] Bojar, Ondrej, Buck, Christian, Federmann, Christian, Haddow, Barry, Koehn, Philipp, Leveling, Johannes, Monz, Christof, Pecina, Pavel, Post, Matt, Saint-Amand, Herve, & others. 2014. Findings of the 2014 Workshop on Statistical Machine Translation. *In: Proceedings of the Ninth Workshop on Statistical Machine Translation (WMT)*.

[Bojar *et al.*, 2015] Bojar, Ondrej, Chatterjee, Rajen, Haddow, Barry, Huck, Matthias, Hokamp, Chris, Koehn, Philipp, Logacheva, Varvara, Monz, Christof, Negri, Matteo, Post, Matt, Scarton, Carolina, Specia, Lucia, & Turchi, Marco. 2015. Findings of the 2015 Workshop on Statistical Machine Translation. *In: Proceedings of the Tenth Workshop on Statistical Machine Translation (WMT)*.

[Bottou, 1991] Bottou, Léon. 1991. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nimes*, **91**(8).

[Bridle, 1990] Bridle, John S. 1990. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. *Pages 227–236 of: Neurocomputing*. Springer.

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

[Collobert *et al.*, 2011] Collobert, Ronan, Weston, Jason, Bottou, Léon, Karlen, Michael, Kavukcuoglu, Koray, & Kuksa, Pavel. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, **12**, 2493–2537.

[de Souza *et al.*, 2014] de Souza, José GC, Politecnica de Valencia, U., Buck, Christian, Turchi, Marco, & Negri, Matteo. 2014. FBK-UPV-UEdin participation in the WMT14 Quality Estimation shared-task. *Pages 322–328 of: Proceedings of the Ninth Workshop on Statistical Machine Translation*.

[Deng, 2014] Deng, Li. 2014. Deep Learning: Methods and Applications. *Foundations and Trends® in Signal Processing*, **7**(3-4), 197–387.

[Devlin *et al.*, 2014] Devlin, Jacob, Zbib, Rabih, Huang, Zhongqiang, Lamar, Thomas, Schwartz, Richard M., & Makhoul, John. 2014. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. *Pages 1370–1380 of: ACL (1)*.

[Duchi *et al.*, 2011] Duchi, John, Hazan, Elad, & Singer, Yoram. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, **12**, 2121–2159.

[Dyer *et al.*, 2010] Dyer, Chris, Lopez, Adam, Ganitkevitch, Juri, Weese, Jonathan, Ture, Ferhan, Blunsom, Phil, Setiawan, Hendra, Eidelman, Vladimir, & Resnik, Philip. 2010. cdec: A Decoder, Alignment, and Learning Framework for Finite-State and Context-Free Translation Models. *In: Proceedings of the ACL 2010 System Demonstrations*.

[Erhan *et al.*, 2010] Erhan, Dumitru, Bengio, Yoshua, Courville, Aaron, Manzagol, Pierre-Antoine, Vincent, Pascal, & Bengio, Samy. 2010. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, **11**, 625–660.

[Finkelstein *et al.*, 2001] Finkelstein, Lev, Gabrilovich, Evgeniy, Matias, Yossi, Rivlin, Ehud, Solan, Zach, Wolfman, Gadi, & Ruppin, Eytan. 2001. Placing search in context: The concept revisited. *Pages 406–414 of: Proceedings of the 10th international conference on World Wide Web*. ACM.

[Goel *et al.*, 2008] Goel, Sharad, Langford, John, & Strehl, Alexander L. 2008. Predictive Indexing for Fast Search. *In: Advances in Neural Information Processing Systems (NIPS)*.

[Goldberg & Levy, 2014] Goldberg, Yoav, & Levy, Omer. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

[Gouws & Søgaard, 2015] Gouws, Stephan, & Søgaard, Anders. 2015. Simple task-specific bilingual word embeddings. *Pages 1386–1390 of: Proceedings of NAACL-HLT*.

[Gouws *et al.*, 2014] Gouws, Stephan, Bengio, Yoshua, & Corrado, Greg. 2014. Bilbowa: Fast bilingual distributed representations without word alignments. *arXiv preprint arXiv:1410.2455*.

[Graham, 2015] Graham, Yvette. 2015. Improving evaluation of machine translation quality estimation. *Pages 1804–1813 of: 53rd Annual Meeting of the Association for Computational Linguistics and Seventh International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*.

[Graham *et al.*, 2014] Graham, Yvette, Baldwin, Timothy, Moffat, Alistair, & Zobel, Justin. 2014. Is Machine Translation Getting Better over Time? *EACL 2014*, 443.

[Hassan & Mihalcea, 2009] Hassan, Samer, & Mihalcea, Rada. 2009. Cross-lingual semantic relatedness using encyclopedic knowledge. *Pages 1192–1201 of: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3*. Association for Computational Linguistics.

[Heafield & Lavie, 2011] Heafield, Kenneth, & Lavie, Alon. 2011. CMU System Combination in WMT 2011. *In: Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*.

[Hermann & Blunsom, 2014] Hermann, Karl Moritz, & Blunsom, Phil. 2014. Multilingual models for compositional distributed semantics. *arXiv preprint arXiv:1404.4641*.

[Hochreiter, 1991] Hochreiter, Sepp. 1991. Untersuchungen zu dynamischen neuronalen Netzen. *Diploma thesis, Institut für Informatik, Technische Universitat, München*.

[Hochreiter, 1998] Hochreiter, Sepp. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, $\mathbf{6}(02)$, 107–116.

[Hochreiter & Schmidhuber, 1997] Hochreiter, Sepp, & Schmidhuber, Jürgen. 1997. Long short-term memory. *Neural computation*, $\mathbf{9}(8)$, 1735–1780.

[Hokamp *et al.*, 2014] Hokamp, Chris, Calixto, Iacer, Wagner, Joachim, & Zhang, Jian. 2014. Target-centric features for translation quality estimation. *Pages 329–334 of: Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland USA: Association for Computational Linguistics.

[Hornik *et al.*, 1989] Hornik, Kurt, Stinchcombe, Maxwell, & White, Halber. 1989. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, $\mathbf{2}$, 359–366.

[Huang *et al.*, 2012] Huang, Eric H, Socher, Richard, Manning, Christopher D, & Ng, Andrew Y. 2012. Improving word representations via global context and multiple word prototypes. *Pages 873–882 of: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics.

[Ivakhnenko & Lapa, 1966] Ivakhnenko, Alekse Grigorevich, & Lapa, Valentin Grigorevich. 1966. *Cybernetic predicting devices*. Tech. rept. DTIC Document.

[Karakos *et al.*, 2008] Karakos, Damianos, Eisner, Jason, Khudanpur, Sanjeev, & Dreyer, Markus. 2008. Machine Translation System Combination using ITG-based Alignments. *In: Proceedings of ACL-08: HLT, Short Papers*.

[Klementiev *et al.*, 2012] Klementiev, Alexandre, Titov, Ivan, & Bhattarai, Binod. 2012. Inducing Crosslingual Distributed Representations of Words.

[Koehn, 2005] Koehn, Philipp. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. *In: Proceedings of Machine Translation Summit X*.

[Kreutzer *et al.*, 2015] Kreutzer, Julia, Schamoni, Shigehiko, & Riezler, Stefan. 2015. QUality Estimation from ScraTCH (QUETCH): Deep Learning for Word-level Translation Quality Estimation. *In: Proceedings of the Tenth Workshop on Statistical Machine Translation*.

[Levy & Goldberg, 2014] Levy, Omer, & Goldberg, Yoav. 2014. Neural word embedding as implicit matrix factorization. *Pages 2177–2185 of: Advances in Neural Information Processing Systems*.

[Luong *et al.*, 2014] Luong, Ngoc-Quang, Besacier, Laurent, & Lecouteux, Benjamin. 2014. LIG System for Word Level QE task at WMT14. *Pages 335–341 of: Proceedings of the Ninth Workshop on Statistical Machine Translation*. Baltimore, Maryland USA: Association for Computational Linguistics.

[Luong *et al.*, 2015] Luong, Thang, Pham, Hieu, & Manning, Christopher D. 2015. Bilingual word representations with monolingual quality in mind. *Pages 151–159 of: Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*.

[Mikolov *et al.*, 2013a] Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S., & Dean, Jeff. 2013a. Distributed representations of words and phrases and their compositionality. *Pages 3111–3119 of: Advances in neural information processing systems*.

[Mikolov *et al.*, 2013b] Mikolov, Tomas, Chen, Kai, Corrado, Greg, & Dean, Jeffrey. 2013b. Efficient estimation of word representations in vector space. *In: Proceedings of Workshop at ICLR*.

[Mikolov *et al.*, 2013c] Mikolov, Tomas, Le, Quoc V., & Sutskever, Ilya. 2013c. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

[Mikolov *et al.*, 2013d] Mikolov, Tomas, Yih, Wen-tau, & Zweig, Geoffrey. 2013d. Linguistic Regularities in Continuous Space Word Representations. *In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

[Miller & Charles, 1991] Miller, George A., & Charles, Walter G. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, $\mathbf{6}(1)$, 1–28.

[Ng *et al.*, 2015] Ng, Raymond WM, Shah, Kashif, Specia, Lucia, & Hain, Thomas. 2015. A study on the stability and effectiveness of features in quality estimation for spoken language translation. *In: Sixteenth Annual Conference of the International Speech Communication Association*.

[Padó, 2006] Padó, Sebastian. 2006. *User's guide to `sigf`: Significance testing by approximate randomisation*.

[Papineni *et al.*, 2002] Papineni, Kishore, Roukos, Salim, Ward, Todd, & Zhu, Wei-Jing. 2002. BLEU: a method for automatic evaluation of machine translation. *Pages 311–318 of: Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics.

[Pedregosa *et al.*, 2011] Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, & others. 2011. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, $\mathbf{12}$, 2825–2830.

[Pennington *et al.*, 2014] Pennington, Jeffrey, Socher, Richard, & Manning, Christopher. 2014. GloVe: Global Vectors for Word Representation. *In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[Ranzato *et al.*, 2007] Ranzato, Marc Aurelio, Huang, Fu Jie, Boureau, Y-Lan, & LeCun, Yann. 2007. Unsupervised learning of invariant feature hierarchies with applications to object recognition. *Pages 1–8 of: IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

[Reese *et al.*, 2010] Reese, Samuel, Boleda, Gemma, Cuadros, Montse, Padró, Lluís, & Rigau, German. 2010. Wikicorpus: A Word-Sense Disambiguated Multilingual Wikipedia Corpus. *In: Proceedings of 7th Language Resources and Evaluation Conference (LREC)*.

[Rosenblatt, 1985] Rosenblatt, Frank. 1985. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**(6), 386–408.

[Rumelhart *et al.*, 1986] Rumelhart, David E., Hinton, Geoffrey E., & Williams, Ronald J. 1986. Learning representations by back-propagating errors. *Nature*, **323**.

[Sahlgren, 2006] Sahlgren, Magnus. 2006. The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces.

[Saxe *et al.*, 2011] Saxe, Andrew, Koh, Pang W, Chen, Zhenghao, Bhand, Maneesh, Suresh, Bipin, & Ng, Andrew Y. 2011. On random weights and unsupervised feature learning. *In: Proceedings of the 28th International Conference on Machine Learning (ICML)*.

[Schamoni & Riezler, 2015] Schamoni, Shigehiko, & Riezler, Stefan. 2015. Combining Orthogonal Information in Large-Scale Cross-Language Information Retrieval. *In: Proceedings of the 38th Annual ACM SIGIR Conference (SIGIR)*.

[Schamoni *et al.*, 2014] Schamoni, Shigehiko, Hieber, Felix, Sokolov, Artem, & Riezler, Stefan. 2014. Learning Translational and Knowledge-based Similarities from Relevance Rankings for Cross-Language Retrieval. *In: Proceedings of the Association of Computational Linguistics (ACL)*.

[Schmidhuber, 2015] Schmidhuber, Jürgen. 2015. Deep learning in neural networks: an overview. *Neural networks: the official journal of the International Neural Network Society*, **61**, 85–117.

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

[Schwenk, 2007] Schwenk, Holger. 2007. Continuous space language models. *Computer Speech & Language*, **21**(3), 492–518.

[Shah & Specia, 2014] Shah, Kashif, & Specia, Lucia. 2014. Quality estimation for translation selection. *In: Proceedings of the 17th Annual Conference of the European Association for Machine Translation, Dubrovnik, Croatia*.

[Shah *et al.*, 2013] Shah, Kashif, Cohn, Trevor, & Specia, Lucia. 2013. An investigation on the effectiveness of features for translation quality estimation. *In: Proceedings of the Machine Translation Summit*.

[Shah *et al.*, 2015a] Shah, Kashif, Ng, Raymond WM, Bougares, Fethi, & Specia, Lucia. 2015a. Investigating Continuous Space Language Models for Machine Translation Quality Estimation.

[Shah *et al.*, 2015b] Shah, Kashif, Logacheva, Varvara, Paetzold, G., Blain, Frederic, Beck, Daniel, Bougares, Fethi, & Specia, Lucia. 2015b. SHEF-NN: Translation Quality Estimation with Neural Networks. *Pages 342–347 of: Proceedings of the Tenth Workshop on Statistical Machine Translation*.

[Shang *et al.*, 2015] Shang, Liugang, Cai, Dongfeng, & Ji, Duo. 2015. Strategy-Based Technology for Estimating MT Quality. *Pages 348–352 of: Proceedings of the Tenth Workshop on Statistical Machine Translation*.

[Snover *et al.*, 2006] Snover, Matthew, Dorr, Bonnie, Schwartz, Richard, Micciulla, Linnea, & Makhoul, John. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. *Pages 223–231 of: Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA)*.

[Son *et al.*, 2012] Son, Le Hai, Allauzen, Alexandre, & Yvon, François. 2012. Continuous space translation models with neural networks. *Pages 39–48 of: Proceedings of the 2012 conference of the north american chapter of the association for computational linguistics: Human language technologies*. Association for Computational Linguistics.

[Soricut & Echihabi, 2010] Soricut, Radu, & Echihabi, Abdessamad. 2010. Trustrank: Inducing trust in automatic translations via ranking. *Pages 612–621 of: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

[Specia *et al.*, 2010] Specia, Lucia, Raj, Dhwaj, & Turchi, Marco. 2010. Machine translation evaluation versus quality estimation. *Machine Translation*, **24**(1), 39–50.

[Specia *et al.*, 2013] Specia, Lucia, Shah, Kashif, De Souza, José GC, & Cohn, Trevor. 2013. QuEst-A translation quality estimation framework. *Pages 79–84 of: ACL (Conference System Demonstrations)*. Citeseer.

[Specia *et al.*, 2015] Specia, Lucia, Paetzold, G., & Scarton, Carolina. 2015. Multi-level Translation Quality Prediction with QuEst++. *Pages 115–120 of: 53rd Annual Meeting of the*

Association for Computational Linguistics and Seventh International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing: System Demonstrations.

[Sundermeyer *et al.*, 2014] Sundermeyer, Martin, Alkhouli, Tamer, Wuebker, Joern, & Ney, Hermann. 2014. Translation Modeling with Bidirectional Recurrent Neural Networks. *Pages 14–25 of: EMNLP*.

[Sánchez & Forcada, 2015] Sánchez, Miquel Espla-Gomis Felipe, & Forcada, Martınez Mikel L. 2015. UAlacant word-level machine translation quality estimation system at WMT 2015. *EMNLP 2015*, 309.

[Temnikova *et al.*, 2015] Temnikova, Francisco Guzmán Ahmed Abdelali Irina, Sajjad, Hassan, & Vogel, Stephan. 2015. How do Humans Evaluate Machine Translation. *EMNLP 2015*, 457.

[Turchi *et al.*, 2015] Turchi, Marco, Negri, Matteo, & Federico, Marcello. 2015. MT Quality Estimation for Computer-assisted Translation: Does it Really Help? *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, **2 (Short Papers)**(July), 530–535.

[Turian *et al.*, 2010] Turian, Joseph, Ratinov, Lev, & Bengio, Yoshua. 2010. Word representations: a simple and general method for semi-supervised learning. *Pages 384–394 of: Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics.

[Turney & Pantel, 2010] Turney, Peter D., & Pantel, Patrick. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, **37**(1), 141–188.

[Ueffing *et al.*, 2003] Ueffing, Nicola, Macherey, Klaus, & Ney, Hermann. 2003. Confidence Measures for Statistical Machine Translation. 394–401.

[Uszkoreit & Lommel, 2013] Uszkoreit, Hans, & Lommel, Arle. 2013. *Multidimensional Quality Metrics: A New Unified Paradigm for Human and Machine Translation Quality Assessment*.

[Weston *et al.*, 2008] Weston, J., Rattle, F., & Collobert, R. 2008. Deep Learning via Semi-Supervised Embedding. *In: International Conference on Machine Learning, ICML*.

[Wisniewski *et al.*, 2014] Wisniewski, Guillaume, Pécheux, Nicolas, Allauzen, Alexandre, & Yvon, François. 2014. Limsi Submission for WMT'14 QE task. *Pages 348–354 of: Proceedings of The ninth Workshop on Statistical Machine Translation*.

[Wolf *et al.*, 2014] Wolf, Lior, Hanani, Yair, Bar, Kfir, & Dershowitz, Nachum. 2014. Joint word2vec networks for bilingual semantic representations. *International Journal of Computational Linguistics and Applications*, **5**(1), 27–44.

[Zeiler, 2012] Zeiler, Matthew D. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

[Zou *et al.*, 2013] Zou, Will Y., Socher, Richard, Cer, Daniel M., & Manning, Christopher D. 2013. Bilingual Word Embeddings for Phrase-Based Machine Translation. *Pages 1393–1398 of: EMNLP*.

[Řehůřek & Sojka, 2010] Řehůřek, Radim, & Sojka, Petr. 2010. Software Framework for Topic Modelling with Large Corpora. *Pages 45–50 of: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA.