Karlsruhe Institute of Technology

Improvement of the Translation of Named Entities in Neural Machine Translation

Master's Thesis of

Maciej Modrzejewski

at the Department of Informatics
Institute for Anthropomatics and Robotics (IAR)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	Prof. Dr. Tamim Asfour
Advisors:	Dr. Thanh-Le Ha
	Dr. Miriam Exel, SAP SE
	Dr. Bianka Buschbeck, SAP SE

14. November 2019 – 26. June 2020

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

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

Karlsruhe, 26.06.2020

. . . . . .

(Maciej Modrzejewski)

Abstract

Neural Machine Translation (NMT) systems have achieved better translation performance compared to the Statistical Machine Translation (SMT) systems in the recent years and are perceived as the state-of-the-art solution in Machine Translation (MT). Nevertheless, the translation quality of the NMT output is often poor for words occurring infrequently in the training corpus or for words with multiple meanings due to their high ambiguity.

One group of such words are named entities (NEs). Their correct translation poses a challenge for NMT systems. In general, conventional NMT systems are expected to translate named entities by learning complex linguistic aspects and ambiguous terms from the training corpus only. When faced with named entities, NMT systems are found to be occasionally distorting location, organization or person names and even sometimes ignoring low-frequency proper names altogether.

Recent approaches in NMT successfully enrich the source language sentences by adding linguistic features into the neural network input with the use of source factors. Word level factors may carry linguistic information (part-of-speech tags, lemmas or morphosyntactic labels), yet may be also used to augment the source sentence with other types of information, e.g. to denote named entities. The incorporation of word features into the neural network input in the context of named entities is a promising approach. The tagging of NEs in the source sentence may support the networks in capturing named entities better, decreasing their ambiguity and thus enhancing their learning process. This thesis aims at exploring its potential and studies methods incorporating Named Entity Recognition (NER) into NMT with the aim to improve named entity translation. It proposes an annotation method that integrates named entities and inside-outside-beginning (IOB) tagging into the neural network input with the use of source factors.

In our experiments, we focus on three named entity classes: organization, location and person. We investigate how the granularity of named entity class labels influences named entity translation quality. Further, we execute an extensive evaluation of the MT output assessing the influence of our annotation method on named entity translation. Finally, we discuss our findings based on translation examples.

Our experiments on English→German and English→ Chinese show that just by including different named entity classes and IOB tagging, we can increase the BLEU score by around 1 point using the standard test set from WMT2019 and achieve up to 12% increase in NE translation rates over a strong baseline. Furthermore, we also illustrate that our annotation technique does not result in a poor translation performance in the scenario where no named entities are present.

Zusammenfassung

Neuronale maschinelle Übersetzungssysteme (NMT) haben eine bessere Übersetzungsleistung im Vergleich zum statistischen Ansatz in den letzten Jahren erzielt und werden als die Lösung auf dem modernsten Stand der Technik in der maschinellen Übersetzung wahrgenommen. Trotzdem ist die Übersetzungsqualität ihrer Ausgabe häufig mangelhaft für Wörter, die selten im Trainingskorpus vorkommen oder für Wörter mit mehreren Bedeutungen aufgrund ihrer hohen Mehrdeutigkeit.

Eigennamen machen eine Gruppe solcher Wörter aus. Ihre korrekte Übersetzung stellt eine Herausforderung für die neuronale maschinelle Übersetzungssysteme dar. Im Allgemeinen wird es von den konventionellen neuronalen maschinellen Übersetzungssystemen erwartet, die Eigennamen zu übersetzen, indem sie komplexe sprachliche Aspekte und mehrdeutige Begriffe nur aus den Trainingsdaten lernen. Wenn NMT-Systeme den Eigennamen begegnen, stellt sich heraus, dass sie vereinzelt Standort-, Organisations- oder Personennamen verzerren und gelegentlich sogar niederfrequente Eigennamen insgesamt ignorieren.

In letzter Zeit konzentrieren sich die Ansätze im NMT Umfeld auf die Bereicherung der Sätze aus der Quellsprache durch das Hinzufügen sprachlicher Merkmale in die Eingabe eines neuronalen Netzwerks unter Zuhilfenahme von *Source Factors*, zu Deutsch quellsprachlichen Faktoren. Faktoren auf Wortebene können sprachliche Informationen enthalten (z.B. die Wortart (part-of-speech) Tags, Lemmas oder morphosyntaktische Markierungen), können jedoch auch dazu verwendet werden, um den Quellensatz mit anderen Informationsarten zu ergänzen, z.B. um Eigennamen zu kennzeichnen. Die Einbeziehung von Wortmerkmalen in die Eingabe eines neuronalen Netzwerks im Kontext von Eigennamen ist ein vielversprechender Ansatz. Durch das Tagging von Eigennamen im Quellensatz können die Netzwerke dabei unterstützt werden, Eigennamen besser zu erfassen, ihre Mehrdeutigkeit zu verringern und dadurch zur Verbesserung ihres Lernprozesses beitragen. Diese Thesis setzt sich als Ziel dieses Potenzial zu erforschen und untersucht Methoden, die Eigennamenerkennung mit dem Ziel, die Übersetzung von Eigennamen zu verbessern, einbeziehen. Sie schlägt eine Annotationsmethode vor, die die Eigennamen und Inside-Outside-Beginning (IOB) Tagging miteinander in die Eingabe eines neuronalen Netzwerks unter Zuhilfenahme von quellsprachlichen Faktoren integriert.

In unseren Experimenten konzentrieren wir uns auf drei Eigennamenklassen: Organisation, Standort und Person. Wir untersuchen auch, wie die Granularität der Eigennamenklassen ihre Übersetzungsqualität beeinflusst. Darüber hinaus führen wir eine umfassende Bewertung der Ausgabe aus dem Übersetzungssystem durch, um den Einfluss unserer Annotationsmethode auf die Übersetzung von Eigennamen zu bewerten. Abschließend diskutieren wir unsere Ergebnisse anhand von Übersetzungsbeispielen.

Unsere Experimente von Englisch→Deutsch und Englisch→Chinesisch zeigen, dass wir den BLEU-Score um etwa 1 Punkt auf dem Standard WMT2019 Testdatensatz erhöhen

können, indem wir verschiedene Eigennamenklassen und IOB-Tags miteinander kombinieren. Wir erreichen bis zu 12% Verbesserung in Eigennamenübersetzungsraten über ein starkes Basismodell. Darüber hinaus veranschaulichen wir auch, dass unsere Annotationsmethode im Szenario, wo keine Eigennamen auftreten, zu keinem Qualitätseinbußen in der Übersetzungsleistung führt.

Contents

List of Figures

List of Tables

1. Introduction

Thanks to the rise of World Wide Web-based technologies our world becomes more and more interconnected. With an increasing level of digitization, there is an explosion of information in the form of news, articles, social media posts, and others forms of communication. Overall, the last decade has witnessed a massive explosion of information in life science. The productivity of companies, regions and nations depends largely on their ability to create and process knowledge-based information effectively.

As information becomes an omnipresent element of our lives, there is a stronger need to communicate with one another across various nations. At the current time, there are approx. 6,500 developed languages, 91 of which have more than 10 million active speakers (Eberhard et al., 2020). A further intensification of international cooperation requires overcoming language barriers. Inevitably, this requires the ability to understand multiple languages. Regrettably, the human capabilities of learning multiple foreign languages are constrained. This drives the need for automated translation solutions.

Interestingly, the need for translation is not limited to face-to-face interactions only but also is required to translate unstructured documents such as, e.g., governmental texts, product descriptions, business transactions texts and others. Reliable and on the fly translation is necessary to promote cultural exchange, collaborative, intercultural research and globalized trade. As a result, there is a high demand for fast, economical and reliable machine translation systems to facilitate information exchange.

The research field of Machine Translation (MT) investigates the approaches to translating text from one natural human language (source language) into another (target language) with no human translator involved. It may be categorized as a sub-field of computational linguistics that takes from a broad spectrum of other disciplines such as linguistics, computer science, information theory, artificial intelligence, and statistics.

In its early days, MT has been criticized for bad quality: lack of fluency and intelligibility, low accuracy and inappropriate style. However, thanks to an intensive research in this area, we have been witnessing great progress in MT quality. Its quality is still lower than human translation, but it does not imply that no good practical uses exist. On the contrary, MT is nowadays growing in popularity and its output is widely consumed by the translation industry: in informal settings (e.g. offered as a plugin on a website or in a messaging chat) and by professional translators. Depending on the scenario, a varying level of translation accuracy is expected. In informal scenarios users wish to receive a fast, somewhat accurate translation. In professional scenarios a very high quality is expected. Currently MT systems do not provide such quality. Therefore, the output of a MT is additionally revised by a human translator to ensure it. In some cases, with appropriate controls on the language and the domain of the input texts, translations can be machine translated that are of high quality requiring no revision. The urge to deliver such solutions producing accurate and fluent translations drives the research in the area of MT.

1.1. Motivation

Neural Machine Translation (NMT) has recently shown promising results, replacing Statistical Machine Translation (SMT) as state-of-the-art approach to machine translation. Technological advances, such as sequence-to-sequence (Sutskever et al., 2014), attention mechanism (Luong et al., 2015) and Transformer networks (Vaswani et al., 2017) greatly contributed to improving accuracy and fluency of machine translation. Within a few years after its first introduction, NMT is at the current time used commercially in productive systems (e.g. Crego et al., 2016).

In order to achieve the goal of creating an NMT system which translates with an identical quality as a human translator, the ongoing research focuses on the elimination of the existing deficiencies of NMT. Koehn and Knowles (2017) outline six main challenges of NMT. In their work, they state the following: "both SMT and NMT systems actually have their worst performance on words that were observed a single time in the training corpus, (...), even worse than for unobserved words. The most common categories across both are named entity (including entity and location names) and nouns". There is, in fact, an intensive research ongoing which aims to improve the translation of named entities in the context of NMT (e.g. Li et al., 2018b; Ugawa et al., 2018; Li et al., 2018a; Yan et al., 2018, and others). Furthermore, there is a workshop series organized by the Association of Computer Linguistics (ACL) specifically dedicated to the research in the area of named entity translation.In light of this deficiency of NMT, this work conducts research of methods which aim to improve the translation of named entities in NMT.

Named entities are the phrases in human languages that explicitly link notations in languages to the entities existing in the real world (Wu et al., 2008). The concept of a "named entity" has been first introduced in the 6th Message Understanding Conference (MUC-6, Grishman and Sundheim (1996)). The aim of this conference was to recognize and subsequently classify named entities into a category (e.g. their type). This task is is referred to as Named Entity Recognition (NER). NER is a research field and focuses on the automatic identification and classification of selected types of named entities in unstructured documents (Nadeau and Sekine, 2007). NER systems are often adopted as an early annotation step in many Natural Language Processing (NLP) pipelines for applications such as question answering, information retrieval and machine translation.

The translation of named entities is challenging because new phrases in the form of personal names, organizations, locations, product names, and monetary expressions appear on a daily basis and many named entities are domain specific, not to be found in bilingual dictionaries. Additionally, the lexical and syntactic ambiguity of named entities creates an obstacle during translation. For example, the word "France" (in English) may refer to a name of a person or the name of a country and depending on the target language its translation necessitates a different inflection.

Improving named entity translation is important due to a number of reasons. First, the correct translation of named entities constitutes a key element to the correct interpretation of scientific, corporate or governmental texts where homogeneous understanding of the handled material is required. Moreover, translation systems and cross-language information retrieval applications depend on their correct translation as a significant number of users' requests have been found to contain them (Jiang et al., 2007). Furthermore, named

entities carry more semantic information than regular content or functional words and have, therefore, a higher information utility. As a result, their mistranslation leads to a higher information loss and impedes the correct understanding of translated texts more severely than a mistranslation of a regular word (Huang, 2005). Finally, a majority of out-of-vocabulary terms are named entities. Consequently, their incorrect or missing translation has a considerable impact on the information retrieval effectiveness and machine translation quality (Wu et al., 2008).

Incorporation of word features into the source sentence Neural Machine Translation is based on a sequence-to-sequence learning approach that interprets sentences as sequences of generic tokens. As such, it does not explicitly exploit external sources providing potentially beneficial linguistic information. Therefore, the question arises whether providing such information, e.g. in form of word features, can help enhancing translation quality.

We analyze this matter in the context of named entity translation. Conventional Neural Machine Translation systems (e.g. Yonghui et al., 2016; Zhou et al., 2016) are expected to translate named entities by learning complex linguistic aspects and ambiguous terms from the training corpus only. There is, however, no guarantee that a NMT system can capture this information and produce a proper translation in all cases, especially for those terms which do not occur very often in the training corpus or are ambiguous. When faced with named entities, NMT systems are found to be occasionally distorting location, organization or person names and even sometimes ignoring low-frequency proper names altogether (Koehn and Knowles, 2017).

Recently Sennrich and Haddow (2016) successfully enriched the source language sentences by adding linguistic features into the neural network input. They find that adding morphological features, part-of-speech tags, and syntactic dependency labels as input features improves translation quality. Their main innovation over the standard encoder-decoder architecture is the ability to represent the encoder input as a combination of features (source factors) which are subsequently concatenated or added to the embedding vector.

In general, a factor refers to "a type of additional word-level information" (Koehn and Hoang, 2007). We define source factors as any type of additional word-level information incorporated into the source sentence exclusively. Word level factors may carry linguistic information, for instance, part of speech tags, lemmas or morphosyntactic labels as in the work of Sennrich and Haddow (2016). However, they may be also used to augment the source sentence with other types of information, e.g. to denote named entities. Generally speaking, factors could be any kind of automatically derivable information that is representable at the word level. External tools, such as e.g. a NER system, may be used to incorporate the annotations into the training corpus and at inference time.

1.2. Research Objective

The incorporation of word features into the neural network input in the context of named entities is a promising approach. This thesis aims at exploring its potential and studies methods incorporating Named Entity Recognition (NER) into NMT with the aim to improve

named entity translation. The NER system acts as an external source of information and its output is used to create word features.

This work explores an annotation method that integrates named entities and inside-outside-beginning (IOB) (Ramshaw and Marcus, 1999) tagging into the neural network input with the use of source factors. In our experiments, we focus on three most common and well-researched named entity classes: Organization, Location and Person. We also investigate how the granularity of named entity class labels influences named entity translation quality. Further, we execute an extensive evaluation of the MT output assessing the influence of our annotation method on named entity translation. Finally, we discuss our findings based on translation examples.

Our experiments on English→German and English→Chinese show that by just including different named entity classes and IOB tagging, we can increase the BLEU score by around 1 point using the standard test set from WMT2019 and achieve up to 12% increase in NE translation rates over a strong baseline. Furthermore, we also illustrate that our annotation technique does not result in a poor translation performance in the scenario where no named entities are present.

1.3. Thesis Outline

The rest of the thesis is structured as follows:

- **Chapter 2, Theoretical Foundations.** We first introduce the essential theoretical foundations regarding this work in the context of NMT.
- **Chapter 3, Related Work.** This chapter outlines previous approaches to named entity translation with a special focus on named entity transliteration as well as statistical and Deep Learning approaches.
- **Chapter 4, System Description.** This chapter outlines the research questions this thesis aims at resolving as well as provides a list of experiments executed in the course of working on this thesis. Additionally, it presents algorithms developed to generate the annotated data.
- **Chapter 5, Selection of the Named Entity Recognition System.** This chapter focuses on outlining the decision process to determine the most suitable Named Entity Recognition system, later used for data annotation.
- **Chapter 6, Evaluation.** This chapter illustrates the evaluation of the experiments and provides answers to the research questions. In addition, it outlines the results of the BLEU scores on different test sets as well as presents an extensive in-depth analysis with the focus on the named entity translation quality. Furthermore, it describes the results of a final human evaluation and provides translation examples.
- **Chapter 7, Conclusion and outlook.** This last chapter concludes this thesis by summarizing its findings and proposes directions for further research in the area of named entity translation.

2. Theoretical Foundations

This chapter provides theoretical fundamentals on which the following work is based on. It starts by outlining the fields of Natural Language Processing and Information Extraction. Later, it continues, in Section 2.3, with providing information about Named Entity Recognition. In detail, the definitions of named entities together with most common NE categories are presented. In the second part of this section, approaches to recognize named entities in texts and the methods to evaluate named entity recognition rates are outlined.

Section 2.4 provides an extensive overview of Neural Machine Translation. It describes word embeddings, encoder-decoder network architecture, the attention mechanism, the training optimization function, the decoding problem and methods to handle large vocabularies. As a next step, Recurrent Neural Machine Translation is presented. Section 2.7 illustrates the notion of self-attention and describes the architecture of the Transformer network. Finally, source factors and the methods to evaluate Machine Translation output are outlined.

2.1. Natural Language Processing

Natural Language Processing (NLP) is concerned with the use of computational methods to process and analyze spoken or written form of free text which acts as a mode of communication commonly used by humans (Assal et al., 2011; Singh, 2018). One of the main goals of NLP is to derive a simpler representation (more computer-readable) of the syntax and semantics from textual information (Collobert et al., 2011). NLP tasks can be divided into two groups:

- **syntactic**: At this level, grammatical rules are applied to determine the basic building blocks of the underlying text. Methods include the segmentation of words into statements, word tokenization, assignment of labels to each token in the form of its part of speech, also referred to as Part-of-speech-Tagging (POS-Tagging),

- **semantic**: At this level, the extraction of semantic information takes place to derive a meaningful representation of words, phrases and sentences. This includes the detection of named entities, identification of positive or negatives sentiments (including negation and uncertainty), machine translation, relation extraction and others (Nadkarni et al., 2011).

2.2. Information Extraction

Information Extraction (IE) combines high and low level aspects of Natural Language Processing. The objective of an IE system is to automatically extract structured information (e.g. entities, relationships between entities, and attributes describing entities) from unstructured sources intended for human use. Later, this information is converted into a structured representation suitable for computer-based storage, processing, and retrieval (Sarawagi et al., 2008; Wimalasuriya and Dou, 2010). The input to an IE system is a collection of unstructured or semi-structured documents, although in general it can be applied to other types such as images, audio recordings or videos as well.

The output and simultaneously the main objective of IE is to identify meaningful pieces of information and to produce a computer-suitable representation of the relevant information from the input document according to previously defined criteria. Such format of data can be used to populate databases that require more structured input.

The most recent approaches (Singh, 2018) to successful extraction of information are:

- **Pattern matching**: This method focuses on checking if in a given sequence of tokens certain extraction patterns occur. This is achieved with the use of regular expressions. These patterns can be easily matched directly with the given input text. As a result, a matched text, which corresponds to an occurrence of a particular entity, is extracted. The drawback of this technique materializes when searching for an entity e.g. name of a location, organization etc., which cannot be easily defined by a regular expression. Despite its apparent limitations, this approach is widely used in practice, e.g. in Muslea et al. (1999).

- **Gazetteer-based approach**: This approach proposes using an external knowledge source (e.g. a pre-defined list) to match chunks of text onto names and entities. Such lists are called a gazette or a gazetteer. Gazetteers also further provide a non-local model for resolving multiple names to the same entity, provided that this entity has a finite number of possible values. This approach requires either creating hand-crafted name lexicons or obtaining a gazette from the corpus or another external source. The limitations of this approach lie in preparing complete and accurate gazette.

- **Machine Learning-based approach**: In this approach, Machine Learning (ML)-based algorithms learn the IE patterns directly from training data by generalizing from a given set of examples, also referred to as ground truths. ML models are created on augmented training data in which all occurrences of named entities of interest are annotated. The use of syntactical linguistic features (POS tagging, word position, capitalization) coupled with the correct placement of the corresponding boundary tags is crucial to meaningful data annotation.

2.3. Named-Entity Recognition

Named Entity Recognition (NER) is a prominent subtask of Information Extraction. It focuses on the automatic identification and classification of selected types of NEs in unstructured documents (Simon, 2017). NER systems are often adopted as an early annotation

step in many Natural Language Processing pipelines for applications such as question answering, information retrieval or topic modeling.

Definitions of named entities

This paragraph presents different definitions of named entities (Nouvel et al., 2016):

- One of the earliest scientific definitions of NEs was coined during the 6th Message Understanding Conference (MUC-6, Grishman and Sundheim (1996)), where NEs were described as "proper names and quantities of interest. Person, organization, and location names were marked as well as dates, times, percentages, and monetary amounts" (Chinchor, 1998).
- During the Conference on Natural Language Learning (CoNLL) in 2003, NEs were defined as: "phrases that contain the names of persons, organizations and locations". Additionally a "miscellaneous" class has been added to cover other NEs which do not belong to the previous three categories (Sang and De Meulder, 2003).
- A more specific definition of NEs has been provided by Meur et al. (2004). In fact, NEs were said to "constitute a particular type of lexical unit referring to a real-world entity in certain specific domains, notably the human, social, political, economic and geographic domains, and which have a name (typically a proper noun or an acronym)".
- A special sub-group of named entities are **compound named entities**. These are named entities that consist of two or more elements (named entities) that exist on their own. For example, componud named entities are: "George Washington Bridge", "Alaska State Troopers" or "Patrick Dwyer".

Named Entity Categories

Although the categories of named entities are predefined, there is a varying understanding of what categories should be perceived as named entities and how broad these categories should become. Several conventions have emerged, and entities are frequently marked up in accordance with the XML style format described in the Message Understanding Conference (Grishman and Sundheim, 1996), where "ENAMEX" tags are used for names, "NUMEX" tags are used for numerical entities, and "TIMEX" tags are used for temporal entities. Table 2.1 presents named entity categories from the MUC conference.

Category	Representatives
ENAMEX	Person, Location, Organization
TIMEX	Temporal Expressions (Date, Time)
NUMEX	Number Expressions (Money, Percent)
MISC	Product

Table 2.1.: Common named entity classes

The categories selected for a particular NER project may be dependable on its requirements. If numerical classification plays a vital role to a particular field, then the categories describing numerical data may need to be more refined. Similarly, if focus lies on the more specific geographical classification, it may be reasonable to classify each location entity as a particular type of location (Zaanen and Mollá, 2007).

Approaches to Named Entity Recognition

Named Entity Recognition can be defined as a word-level tagging problem where each word in a sentence is either mapped to a named entity tag or is classified as a regular common word (Yadav and Bethard, 2019). In the following, we outline the most common approaches to named entity recognition (Li et al., 2020; Yadav and Bethard, 2018);

- **Rule-based approaches**

  Early NER systems were based on handcrafted pattern-based rules, lexicons, orthographic features and ontologies (Callan and Mitamura, 2002; Sekine and Nobata, 2004). This approach proposes to learn extraction rules that rely on linguistic, syntactic, or document format patterns that are homogeneous and consistent across a group of documents. Rules can be created based on syntactic-lexical patterns and domain-specific gazetteers. Rule-based systems work very well provided the lexicon is exhaustive.

  In general, precision tends to be generally high for rule-based NER systems because of the lexicons, but recall may often be low due to domain and language-specific rules and incomplete dictionaries. Another drawback of rule-based NER systems is the need of domain experts for constructing and maintaining the knowledge resources and the fact that system cannot be transferred to other domains (Yadav and Bethard, 2018).

- **Unsupervised Learning Approaches**

  A typical approach of unsupervised learning is clustering. Clustering-based NER systems extract named entities from the clustered groups based on context similarity (Nadeau and Sekine, 2007). The primary concept is that lexical resources, lexical patterns, and statistics computed on a large corpus can be used to infer occurrences of named entities. Examples of studies using unsupervised algorithms for named entity classification are Collins and Singer (1999) and Etzioni et al. (2005).

- **Feature-based Supervised Learning Approaches**

  Supervised machine learning models learn directly from the training data and can be used to replace human-curated rules. This data must be labeled, which implies that to each input there is an expected output defined. The application of supervised learning necessitates feature engineering. Based on these features, many machine learning algorithms have been applied in supervised NER, including Hidden Markov Models, Decision Trees, Maximum Entropy Models, Support Vector Machines and Conditional Random Fields (Li et al., 2020).

  Such systems propose applying a rule system over features vectors, which were defined on the word-level (related to the character makeup of words). They specifically describe

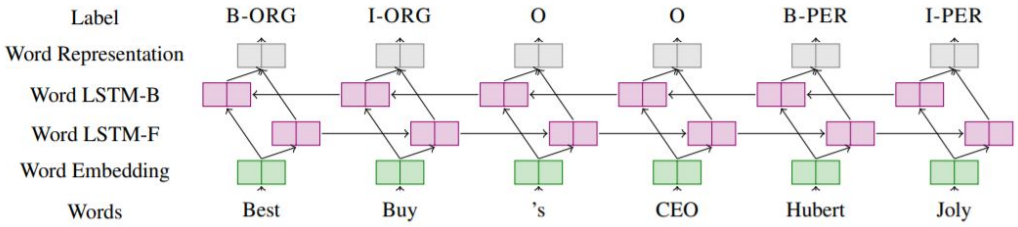

Figure 2.1.: Word-level LSTM-based architecture for NER, from Yadav and Bethard (2018)

word case, punctuation, numerical value and special characters. Additionally, resources in the form of gazetteers, lexicons and dictionaries (referred to as "privileged features") were fed to the classifier to enable lookup techniques.

- **Deep Learning (DL) techniques for NER**

  In recent years, DL-based NER models have become the most prevailing approach to recognize named entities. In comparison to feature-based approaches, deep learning is beneficial in discovering hidden features automatically and thus delivering state-of-the-art results. Deep learning is one of the fields of machine learning that uses the growing volume and availability of data to train models effectively by using increased computational processing power. It focuses on training artificial neural networks which compose of multiple processing layers and learn representations of data with multiple levels of abstraction.

  Contemporary neural architectures for NER can be predominantly classified into categories which depend on their representation of the words in a sentence. For instance, the form of representation may be based on words, characters, other sub-word units or any aggregate of these.

  – **Word level architectures**

    In this architecture, the words of a sentence are given as input to a Recurrent Neural Network (RNN) and each word is represented by its word embedding. Huang et al. (2015) experiment with a variety of Long Short-Term Memory (LSTM) based models for sequence tagging. They present a word LSTM model (Figure 2.1) and showed that adding a Conditional Random Fields (CRF) layer to the top of the word LSTM improved performance, achieving 84.26% F1-score on the English CoNLL 2003 data set.

    Collobert et al. (2011) use a similar architecture to the one shown in Figure 2.1, but a convolution layer is introduced instead of the bidirectional LSTM layer and the output of the convolution layer was forwarded to a CRF layer for the final prediction. The authors achieved 89.59% F1-score on English CoNLL 2003 data set.

  – **Character level architectures**

    In the character level architecture, a sentence is perceived to be a sequence of characters. This sequence is passed through a RNN, predicting labels for each character (Figure 2.2). Character labels are transformed into word labels during a post-processing

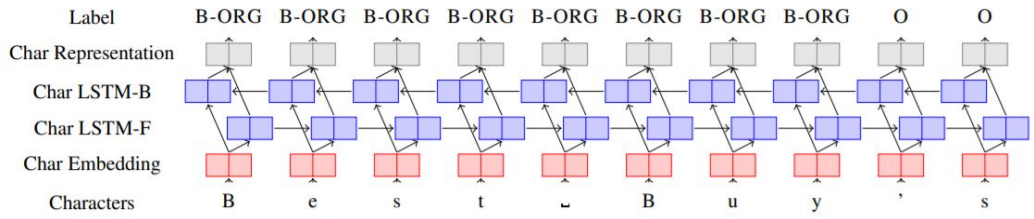

Figure 2.2.: Character-level LSTM-based architecture for NER, from Yadav and Bethard (2018)

step. Character-level representation has been discovered to be effective for exploiting explicit sub-word-level information such as prefix and suffix. Another advantage of character-level representation is that it handles the out-of-vocabulary problem well due to its intrinsic properties. Thus character-based model is capable of inferring representations for unseen words and share information of morpheme-level regularities (Li et al., 2020).

Kuru et al. (2016) propose CharNER, a character-level tagger for language independent NER. CharNER regards a sentence to be a sequence of characters and uses LSTMs to extract meaningful character-level representations. It outputs a tag distribution for each character instead of each word. As a next step, word-level tags are acquired from the character-level tags. Their results demonstrate that taking characters as the primary representation is superior to using words as the basic input unit.

Evaluation of NER systems

The aim of a NER system is to correctly identify a boundary of a named entity and correctly classify its type, simultaneously. The outputs of NER systems are usually evaluated by comparing them against human annotations, also referred to as "ground truths". A True Positive (TP) occurs when a NER system returns an entity which also appears in the ground truth. (Nadeau and Sekine, 2007) present four types of errors which a NER system can produce:

- False Positive (FP) – token which does not appear in the ground truth but is recognized by the NER system (a token has been erroneously classified as an entity)
- False Negative (FN) – entity that is not returned by a NER system but appears in the ground truth (an entity has not been recognized by the system)
- correct type, wrong boundaries – an entity is recognized with the correct type, but its boundaries are wrongly set
- incorrect type and boundaries – both the span and the class of an entity are incorrect

In the following, we discuss the most common evaluation metrics used to assess the quality of NER systems.

$$Precision(P) = \frac{TP}{TP + FP} \qquad Recall(R) = \frac{TP}{TP + FN} \tag{2.1}$$

Precision is defined as the percentage of the system results which are correctly recognized. Recall calculates the percentage of total entities correctly recognized by your system (Li et al., 2020). A measure that integrates precision and recall at once is the harmonic mean of precision and recall, the traditional F-measure or the balanced F-score. Should beta be set to one, the precision and recall are equally important.

$$F_\beta = (1 + \beta^2) * \frac{P * R}{\beta^2 * P + R} \tag{2.2}$$

In order to rank multiple NER systems by their quality of annotation the following two metrics may be used:

- macro-averaged F-score – calculates independently the F-score on different entity types, then takes the harmonic mean of the F-scores
- micro-averaged F-score – sums up the respective false negatives, false positives and true positives across all entity types then applies them to calculate the F-score

The latter can be significantly skewed by the quality of recognizing entities in large classes in the corpus.

2.4. Neural Machine Translation

Neural Machine Translation (NMT) is a fairly recently proposed approach to Machine Translation (MT). In contrast to the Statistical Machine Translation (SMT), which builds a phrase-based translation system, e.g. Koehn (2009), composing of many small subcomponents that are tuned separately, NMT attempts to build and train a single, large artificial neural network that reads a sentence and outputs a correct translation (Stahlberg, 2019).

From a probabilistic point of view, neural machine translation seeks for a parametric model that calculates a conditional probability $P(y|x)$ for a target sentence y given a source sentence x. This goal is common to SMT. Brown et al. (1992) define a frequentist interpretation of $P(y|x)$ as the probability that a human translator would translate x to y.

The general translation problem can be described as finding the most likely target language sentence $\mathbf{y}$ given a source language sentence $\mathbf{x} = (x_1, x_2, ..., x_I)$ of length I (Bahdanau et al., 2015). Using a maximum likelihood approach, $\mathbf{y}$ can be found by solving Equation 2.3. Thanks to the use of the chain rule of conditional probabilities (Goodfellow et al., 2016), the target sentence can be built up incrementally.

$$P(\mathbf{y}|\mathbf{x}) \overset{\text{Chain rule}}{=} \prod_{j=1}^{J} P(y_j|y_1^{j-1}, \mathbf{x}) \tag{2.3}$$

$P(\mathbf{y}|\mathbf{x})$ computes for each target position j the conditional probability $P(y_j|y_{1:j-1}, \mathbf{x})$ of the target word y_j occurring in the translation at position j, given the preceding target words $y_{1:j-1}$ and the source sentence $\mathbf{x}$ (Kalchbrenner and Blunsom, 2013).

2.4.1. Word embeddings

Artificial Neural Networks (ANNs) are designed to learn from numerical data. There is, therefore, a need to represent words in a special input representation which is suitable for the ANNs to be consumed. For this purpose, word embeddings are learned. In essence, they are models that are created to map a set of words or phrases in a vocabulary to vectors of numerical values. They are described in more detail in the second part of this section.

The input unit to a neural network in the area of NMT are words or sub-words. Words are discrete tokens, which can be portrayed by their index in the respective source or target vocabulary (Bengio et al., 2003). In order for neural networks to differentiate between different words, these vectors must be unambiguous and mathematically truly independent in terms of vector distances. One-hot encoding is used to achieved this.

One-hot encoding The aim of One-hot encoding is to represent each word from a vocabulary W by a $|W|$-dimensional vector v with almost all entries set to zero. There are separate vocabularies used for the source and target language, e.g. W_{source} or W_{target}. Only a single position k in the vector v is set to one. It identifies the k^{th} word $w_k \in W$ from the vocabulary. The length of the vector v corresponds with the vocabulary size $|W|$. The ordering of words in the vocabulary must be defined once and remain unchanged during training and inference but is otherwise arbitrary. One-hot encoded vectors are usually the word input for the neuronal networks in NMT. We will denote the one-hot vector of a word index $x \in W$ as $h(x)$.

Embedding matrices Numerical operations with such one-hot encoded vectors for these words would be very inefficient because most values in the one-hot vector are equal 0. As a result, the matrix calculation occurring between the one-hot vector and the first hidden layer will result in an output having mostly 0 values. Therefore, an embedding layer is introduced to greatly improve the efficiency of the network. Embeddings are just like a fully-connected layer with small differences. In detail, its activation functions are linear and a bias vector is usually not used. The dimension of the embedding layer is usually configured to be significantly smaller than the size of the respective input word vocabulary. To compute the embedding layer, the input vector in One-hot encoding, representing the k^{th} vocabulary word, is projected linearly by an embedding matrix E:

$$e(x) = E \cdot h(x), E \in \mathbb{R}^{d \times |W|} \tag{2.4}$$

One of the benefits of using dense and low-dimensional vectors $e(x)$ is computational: a great number of neural network toolkits do not operate well on very high-dimensional, sparse vectors. The main benefit of the dense representations is the generalization power: as certain features may provide similar clues, it is beneficial to provide a representation that is able to capture these similarities (Goldberg, 2016). Learned continuous word representations have the potential of capturing semantic similarity across words as well as their morphological and syntactic features (Collobert and Weston, 2008).

In NMT, embedding matrices are usually trained jointly with the rest of the network using backpropagation and stochastic gradient descent (Rumelhart et al., 1986). In other

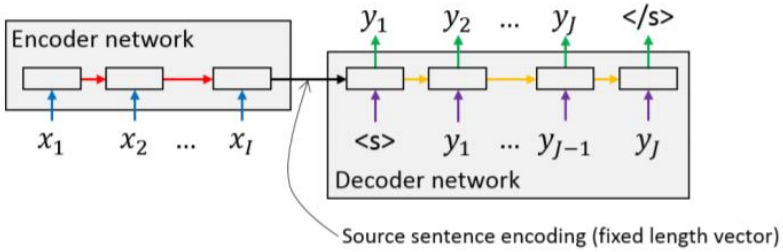

Figure 2.3.: The encoder-decoder architecture of Sutskever et al. (2014). The color coding indicates weight sharing, from Stahlberg (2019).

areas of NLP, pre-trained word embeddings, such as e.g. *word2vec* (Mikolov et al., 2013), trained on unlabelled text have become popular (Collobert et al., 2011).

2.4.2. Encoder-Decoder Networks with Fixed Length Sentence Encodings

Kalchbrenner and Blunsom (2013) developed recurrent continuous translation models (RCTM) I and II. RCTMs map, without a loss of generality, a fixed-length representation of the source sentence to a probability distribution over the sentences in the target language. The development of RCTMs laid the foundation to the rise of to a new family of the so-called encoder-decoder networks. This type of architecture is nowadays the most common in NMT (Stahlberg, 2019). In the current state-of-the art both, encoder and decoder are either based on Recurrent Neural Networks (RNNs) (described in Section 2.5), Convolutional Neural Networks (CNNs) (described in Section 2.6), or on self-attention mechanism (described in Section 2.7).

Encoder-decoder networks are subdivided into an encoder network which reads and encodes a source sentence into a fixed-length vector $c(\mathbf{x})$, and a decoder network which outputs a translation from the encoded vector. Figure 2.3 shows the architecture of an encoder-decoder network developed by Sutskever et al. (2014). First, the source sentence $\mathbf{x}$ is encoded by the encoder network into an internal representation. This representation is later passed to the decoder network. At each time step, the decoder generates a target sentence symbol y_j based on the output y_{j-1} and the decoder's hidden state. The algorithm terminates when the network produces the end-of-sentence symbol `</s>`.

Sutskever et al. (2014) presented one of the first working NMT system operating fully on its own and not relying on any SMT baseline. The main advantage of this approach is the fact that no highly engineered features are used to train the MT system. On the contrary, the architecture is rather simple and the system makes no assumptions on the sequence structure (Stahlberg, 2019).

2.4.3. Attention

One of the challenges for the NMT models, in comparison to the SMT, is the poor translation of longer sentences (Koehn and Knowles, 2017). Sountsov and Sarawagi (2016) refer to it as the "length bias problem". Cho et al. (2014a) suggested that this deficiency is due to the fixed-length source sentence encoding. Sentences with varying length convey different

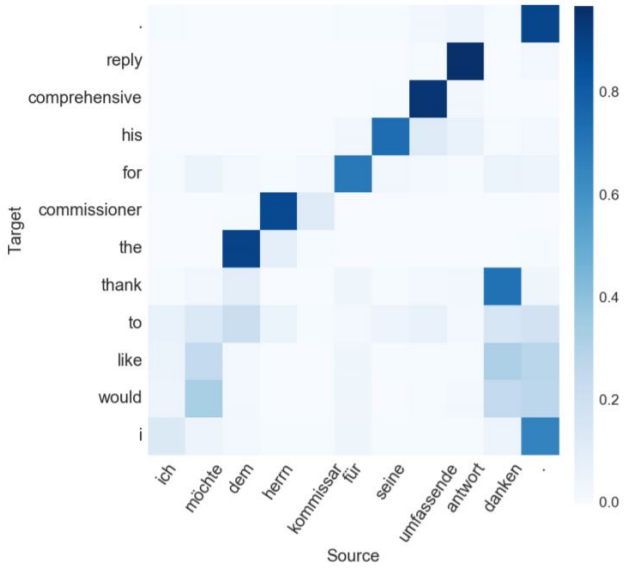

Figure 2.4.: Depiction of the attention paid to the relevant parts of the source sentence for each generated word of a translation example; dark shades of blue indicate high attention weights, from Ghader and Monz (2017)

amounts of information. The longer the sentence, the more information must be stored in the fixed-length context vector. As a result, a fixed-length vector "does not have enough capacity to encode a long sentence with complicated structure and meaning" (Cho et al., 2014a).

Bahdanau et al. (2015) proposed the concept of attention to avoid having a fixed-length source sentence representation. In detail, the encoder does not encode all information in the source sentence into a fixed-length context vector $c(x)$ any more. By contrast, thanks to the attention mechanism, the attentional decoder can pay attention only to the parts of the source sentence which are relevant to produce the next token. The constant context vector $c(x)$ is, as a result, replaced by a series of context vectors $c_j(x)$; one for each time step j. With this novel approach the information can be spread throughout the sequence of annotations, which can be selectively retrieved by the decoder as required. In general, the attention mechanism improves the prediction process by deciding which portion of the source sentence to emphasize at a time (Luong et al., 2015). Nowadays the standard state-of-the-art NMT system consists of an encoder, a decoder and an attention mechanism, which are all trained with maximum likelihood in an sequence-to-sequence fashion (Bahdanau et al., 2015).

Figure 2.6 shows an example of how attention uses the most related source words to produce a target word at each step of the translation.

2.4.4. Training objective

In general, the paramount training objective is to identify optimal parameters values, e.g. the weight of matrices and biases, which minimize a certain error function. Each output of the neural network is compared with the corresponding expected output value from

the training data set (supervised learning). As a next step, the difference between the two before-mentioned values is calculated with the use of the error function formulated for the learning problem at hand. Further, a stochastic gradient descent based algorithm together with back-propagation is used to back-propagate the error (cf. Goodfellow et al. (2016)). The back-propagation is most effective if the objective function is well-differentiable.

Cho et al. (2014b) formulate the training problem in NMT as maximizing the conditional log-likelihood (Equation 2.5). θ denotes the model parameters, e.g., the weight of matrices and biases and each (y_n, x_n) represents a pair of input/output tokens from a sentence pair in the training set.

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(\mathrm{y}_n|\mathrm{x}_n) \tag{2.5}$$

Maximizing Equation 2.5 is equal to minimizing the cross-entropy, which is proportional to the negative log-likelihood of the otherwise alike equation.

ANNs are trained on Graphical Processing Unit (GPUs) for acceleration purposes. GPUs posses a high bandwidth main memory and deep pipelines and are, therefore, well-adapted to efficiently calculate numerous matrix multiplications in parallel.

2.4.5. Neural Machine Translation Decoding

In the beginning of Section 2.4 we describe the translation probability (Equation 2.3). However, the probability $P(\mathbf{y}|\mathbf{x})$ does not provide a method for finding the most probable translation. The task of finding the most probable translation $\tilde{\mathbf{y}}$ for a given source sentence $\mathbf{x}$ is referred to as the decoding or inference problem:

$$\tilde{\mathbf{y}} = \underset{\mathbf{y} \in W_{trg}}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{x}), \tag{2.6}$$

where W_{trg} denotes the target language vocabulary.

NMT decoding is a complex task. The search space is wide in range and grows exponentially with the sequence length. Secondly, decoding strategies are searching for the most probable translation. This does not, unfortunately, imply that the best translation can be found at all times (Stahlberg, 2019). Surprisingly , more exhaustive searches may lead to worse translations (Stahlberg and Byrne, 2019). The following strategies to decoding are used in practice: word sampling from the probability distribution over the target vocabulary at each decoding step, also referred to as "stochastic sampling", greedy decoding and beam search (Gu et al., 2017).

Greedy and beam search are based on the left-to-right factorization of NMT, with partial translation prefixes being scored using the conditional probability: $P(y_j|y_{1:j-1}, \mathbf{x})$. This implies that both approaches work in a time synchronous fashion. In each step j partial hypotheses of (up to) length j are compared against one another, and a subset of them is selected for further analysis in the next time step, the rest is pruned. Selection process is dependent on the traversing technique.

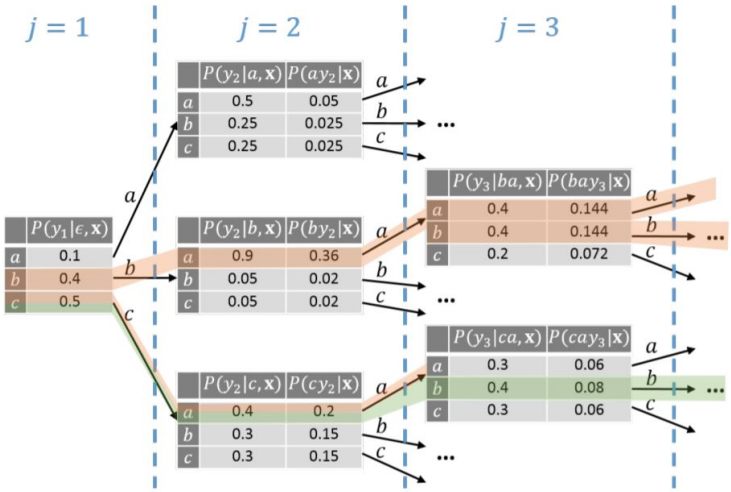

Figure 2.5.: Greedy search (highlighted in green) and beam search (highlighted in orange) with beam size $K = 2$, from Stahlberg (2019)

Greedy search selects the most likely target word y_j generated in each decoding step. This simple approach is vulnerable to the so-called garden-path problem (Koehn, 2017). In certain cases a sequence of words is selected and only later on shall it be realized that initially less probable words suit the context of the full output better. Unfortunately, greedy decoding cannot correct this error later once it is committed to a certain path.

Beam search, on the other hand, considers K most probable translation prefixes to the next time step. In this way, the risk of missing out on the globally most probable translation can be minimized. The K active hypotheses are scored by probability. As a next step, each of these words in the beam is used in the conditioning context for the next word. Due to this conditioning, different word predictions are made for each hypothesis. Later, the score for the partial translation is multiplied with the probabilities from its word predictions. The highest scoring word pairs is selected for the next beam. At each time step only K most probable hypotheses are considered, the rest is pruned. Once the end of sentence sign is reached, the search terminates and the most probable hypothesis is chosen. Usually parameter K is set to 5. Figure 2.5 presents the different in the search strategy between the greedy and beam searches.

Beam search does not guarantee that a translation with higher or equal overall probability than predicted by the greedy search is always found. Beam search is also susceptible to the garden-path problem, although less than the greedy search. Stahlberg and Byrne (2019) state that the beam search may also suffer from a high number of search errors.

2.4.6. Large Vocabularies: Byte Pair Encoding (BPE)

According to Zipf's law (Zipf, 1946) words in a language follow a very unevenly distribution. New words and phrases emerge in the language on a daily basis, e.g. in the form of newly invented words ("website", "retweeting" etc.) and named entities (e.g. names of newly established organizations) (Koehn, 2009).

On the other hand, neural networks are not designed to handle infinitely large vocabularies. Full-blown vocabularies do not fit into the memory of GPUs (García-Martínez et al., 2016). Moreover, a large softmax output layer, calculating a probability distribution over all output words, is computationally very expensive. Therefore, there is an urgent need to reduce the vocabulary size to e.g. between 20.000 to 80.000 words (Koehn, 2009).

One solution to this problem, has been presented by Sennrich et al. (2016). They propose splitting up rare words into subword units allowing thus open-vocabulary translation. Their algorithm uses the logic from Byte Pair Encoding (BPE) compression algorithm (Gage, 1994). BPE is a simple data compression technique, the purpose of which is to replace the most frequent pair of bytes in a sequence with a single, not previously used byte in an iterative fashion. The same logic is applied in the work of Sennrich et al. (2016) to perform word segmentation. In the following, we describe their approach of creating sub-words:

First, each word in the training corpus is split into characters. Original spaces are denoted with a special space character. This is required as no merges should occur between word boundaries. As a next step, most frequent pairs of characters are merged together (e.g. "t" and "h" would be merged to "th" in English). This step is repeated a fixed number of times. Each merge operation results in a new symbol which represents a character n-gram (Sennrich et al., 2016). The number of merge operations k is a hyper-parameter of the algorithm. The final symbol vocabulary size is the sum of the size of the initial vocabulary plus the number of merge operations (Sennrich et al., 2016). The algorithm terminates once k merge operations have been applied.

Languages which share an alphabet (e.g. English and German) benefit from learning BPE while concatenating the involved languages. In this case, applying BPE segmentation to source and target language is called "joint BPE". When BPE is learned jointly (if applicable), the consistency of segmentation increases. Additionally, jointly applied BPE alleviates partially the problem of random character insertion/deletion when transliterating named entities as the vocabularies of the languages are shared.

2.5. Recurrent Neural Machine Translation

The first NMT models using attention are based on RNNs. In Equation 2.3 we defined the conditional probability $P(\mathbf{y}|\mathbf{x})$ of a target sentence $\mathbf{y} = y_1^J$ given a source sentence $\mathbf{x} = x_1^I$. RNNs model this probability as follows (Bahdanau et al., 2015):

$$P(\mathbf{y}|\mathbf{x}) \overset{\text{Chain rule}}{=} \prod_{j=1}^{J} P(y_j|y_1^{j-1}, \mathbf{x}) = \prod_{j=1}^{J} g(y_j|y_{j-1}, s_j, c_j(\mathbf{x})), \tag{2.7}$$

where the function $g(\cdot)$ models the functionality of the decoder network which calculates the distribution for the next target token y_j given the last produced token y_{j-1}, the RNN decoder state $s_j \in \mathbb{R}^n$, and the context vector $c_j(\mathrm{x}) \in \mathbb{R}^m$. The size of the encoder hidden layer is denoted with m, whereas the size of decoder hidden layers with n. The context vector $c_j(\mathbf{x})$ is a distributed representation of the relevant parts of the source sentence (cf. Section 2.4.3).

Encoder: Bidirectional RNN

The encoder network reads the source sentence $\mathbf{x}$ and converts it to a sequence of source sentence annotations $\mathrm{h} = (h_1, ..., h_I)$. Each annotation $h_i \in \mathbb{R}^m$ encodes information about the entire source sentence $\mathbf{x}$ with a special focus on the parts surrounding the i^{th} word of the input sequence.

(Bahdanau et al., 2015) propose a bidirectional RNN (BiRNN, Schuster and Paliwal (1997)) to generate the annotations. A BiRNN consists of two independent RNNs. The forward RNN $\vec{f}$ reads $\mathbf{x}$ in the original order (from x_1 to x_I) and calculates a sequence of forward hidden states $\overrightarrow{h} = (\overrightarrow{h_1}, ..., \overrightarrow{h_I})$. The backward RNN $\overleftarrow{f}$ consumes $\mathbf{x}$ in reversed order (from x_I to x_1) resulting in a sequence of backward hidden states $\overleftarrow{h} = (\overleftarrow{h_1}, ..., \overleftarrow{h_I})$. The next hidden state is computed in the following way:

$$\overrightarrow{h}_i = \overrightarrow{f}(x_i, \overrightarrow{h}_{i-1}) \tag{2.8}$$

$$\overleftarrow{h}_i = \overleftarrow{f}(x_i, \overleftarrow{h}_{i-1}) \tag{2.9}$$

The RNNs $\overrightarrow{f}(\cdot)$ and $\overleftarrow{f}(\cdot)$ are usually Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997a) or Gated Recurrent Unit (GRU) (Cho et al., 2014b) cells. The main advantage of LSTMs and GRUs over RNNs is the fact that they alleviate the vanishing gradient problem (Kolen and Kremer, 2001) and are able to capture long-range dependencies.

The annotation h_i is the concatenation of the hidden states $\overrightarrow{h}_i$ and $\overleftarrow{h}_i$:

$$h_i = [\overrightarrow{h_i^T}; \overleftarrow{h_i^T}]^{\mathrm{T}} \tag{2.10}$$

In this way, the annotation h_i contains the summaries of both the preceding words and the following words.

Alignment model

This sequence of annotations is used by the decoder and the alignment model later to compute the context vector (Bahdanau et al., 2015). The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_j(\mathrm{x}) = \sum_{i=1}^{I} \alpha_{j,i} h_i \tag{2.11}$$

The weights $\alpha_{j,i}$ are determined by the alignment model $a(\cdot)$ which scores how well the inputs around position j and the output at position i match:

$$\alpha_{j,i} = \frac{\exp(a(s_{j-1},\ h_i))}{\sum_{k=1}^{I} \exp(a(s_{j-1},\ h_k))} \tag{2.12}$$

where $a(s_{j-1},\ h_i)$ is a feed-forward neural network which estimates the importance of annotation h_i for producing the j^{th} target token given the current decoder state $s_{j-1} \in \mathbb{R}^n$.

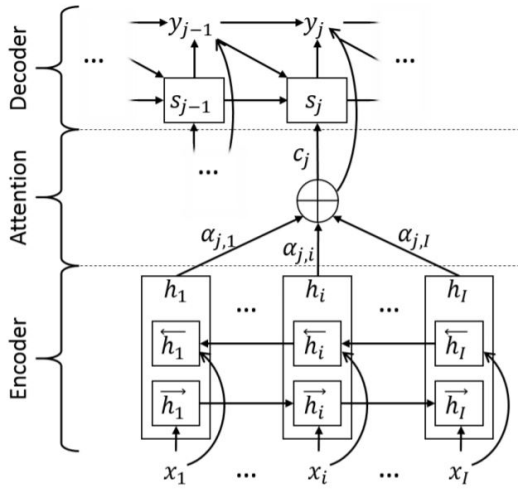

Figure 2.6.: The architecture of the Recurrent Neural Network developed by Bahdanau et al. (2015). Model generates the y_j word given the input sequence $\mathbf{x}$, from Stahlberg (2019)

Decoder

The decoder is also a RNN and it constitutes a Neural Language Model (NLM) for the target language. In Equation 2.7 we stated that $g(\cdot)$ models the functionality of the decoder network and s_j denotes the decoder's state. More specifically, s_j is computed in the following way:

$$s_j = f(s_{j-1},\ y_{j-1},\ c_j) \tag{2.13}$$

where $f(\cdot)$ is modeled by a GRU or LSTM cell. In contrast to the encoder–decoder approach from Section 2.4.2, here the probability is conditioned on a distinct context vector c_j for each target word y_j. Function f may be defined with e.g. the hyperbolic tangent activation function. Its input vectors are projected linearly using the matrices $\mathrm{U}^{\mathrm{dec}}$, $\mathrm{W}^{\mathrm{dec}}$, and C.

$$s_j = f(s_{j-1},\ y_{j-1},\ c_j) \overset{\text{e.g.}}{=} \tanh(U^{\mathrm{dec}} s_{j-1} + W^{\mathrm{dec}} E y_{j-1} + C c_j), \tag{2.14}$$

where E is a target language embedding projection. Using the hidden state s_j, an intermediate representation for the current target word y_t can be computed as follows:

$$\mathrm{y}_j = \phi\ (\mathrm{s}_j\ , y_{t-1}, c_j) \overset{\text{e.g.}}{=} \tanh(\mathrm{Q_s s}_j + \mathrm{Q_y} E y_{t-1} + \mathrm{Q_c c}_j) \tag{2.15}$$

Finally, the output vector o_j representing a probability distribution over all possible target words for the decoder step j can be computed using the following equation:

$$o_j = g(\mathrm{y}_j) \overset{\text{e.g.}}{=} softmax(\mathrm{O y}_j) \tag{2.16}$$

2.6. Convolutional Neural Machine Translation

Convolutional Neural Networks (CNNs) (LeCun and Bengio, 1995) and Time-Delay Neural Networks (TDNNs) (Waibel et al., 1989) are a special variation of feed-forward neural networks which can effectively handle variable-length (time-shifted) sequences and, as a result, capture long-distance dependencies within them. TDNN was the first architecture to achieve shift invariance by sharing weights along temporal dimension. Shift-invariant classification implies that the classifier does not require explicit segmentation prior to classification. This property makes TDNNs suitable for speech processing. For example, TDNNs achieve high performance when applied to far distance speech recognition tasks (e.g. Lang et al., 1990; Dellaert et al., 1996; Peddinti et al., 2015).

In Machine Translation, convolutions are usually one dimensional since MT systems are learning sequences rather than two dimensional images as in computer vision. Using convolutions has some benefits for training NMT models. It reduces sequential computation and CNNs are therefore easier parallelizable on GPUs. Secondly, their hierarchical structure connects distant words via a shorter path than sequential topologies which eases the learning process (Hochreiter et al., 2001). Examples of fully convolutional NMT models (both the encoder and the decoder are convolutional) are ConvS2S (Gehring et al., 2017b) and SliceNet (Kaiser et al., 2017).

2.7. Self-attention-based Neural Machine Translation

In Equation 2.7 we define the conditional probability $P(\mathbf{y}|\mathbf{x})$ with the dependency on the target words y_1^{j-1} until the position $j-1$. In Section 2.5 this dependency is modeled via a recurrent connection which transfers the decoder state back to the next time step. Another possibility to model this dependency is with *self-attention*. Self-attention shortens paths between distant words and decreases the amount of sequential computation (Stahlberg, 2019).

Vaswani et al. (2017) presented a first example of a self-attention network: The Transformer. The Transformer uses attention (cf. Section 2.4.3) in three areas:

1. within the encoder to allow context-sensitive word representations which depend on the whole structure of the source sentence,
2. between the encoder and the decoder, similarly to RNNs (Section 2.5)
3. within the decoder to "remember" the recent translation history.

Lakew et al. (2018) and Vaswani et al. (2017) show that the Transformer networks outperform RNNs in NMT. The reason behind it is that thanks to self-attention, it becomes easier to capture long-distance interdependent features in a given sentence. RNNs, on the other hand, require multiple time steps to accumulate necessary information before it can be linked. With longer distances, it becomes more challenging to capture the information effectively. Finally, Tang et al. (2018) state that the shorter paths are beneficial for learning strong semantic feature extractors and help decrease word sense ambiguity.

In the following, we present the Transformer network as developed by Vaswani et al. (2017).

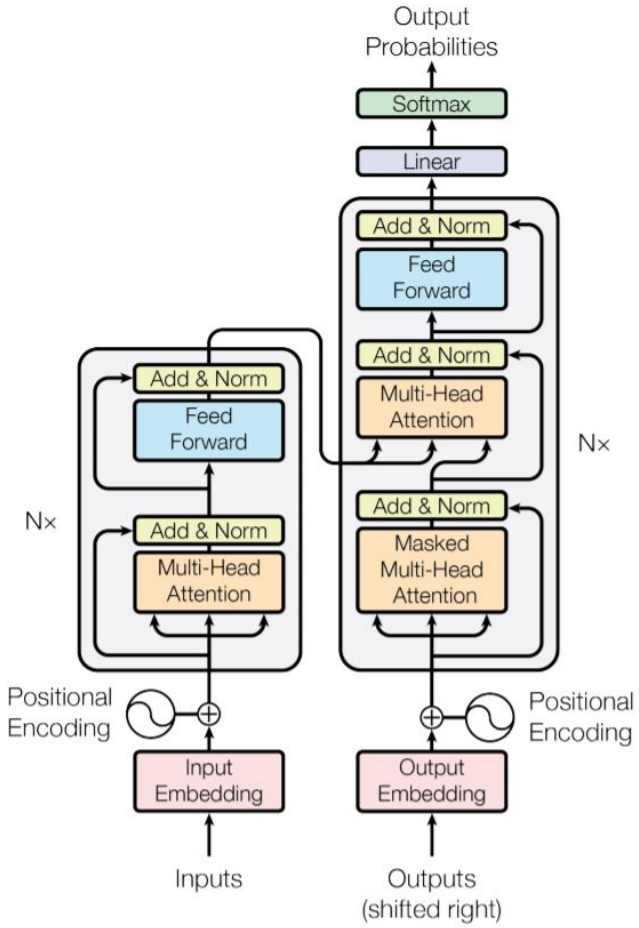

Figure 2.7.: The Transformer - model architecture, from Vaswani et al. (2017)

Positional encoding

As Transformer networks do not use any recurrence or convolution, positional encoding is introduced. Its aim is to provide necessary information about the order of the source sequence. From the input tokens, learned embeddings of dimension d_{model} are generated. Vaswani et al. (2017) use additive encoding which is defined in the following way:

$$PE(pos, 2i) = \sin(pos/10000^{2i/d_{\text{model}}}) \tag{2.17}$$

$$PE(pos, 2i+1) = \cos(pos/10000^{2i/d_{\text{model}}}) \tag{2.18}$$

where pos is the position of a token in the sentence and i is the dimension of the vector.

Encoder and Decoder Stacks

Transformer networks follow the encoder-decoder architecture using stacked self-attention (multi-head attention) followed by a fully-connected feed forward network. Figure 2.7 displays their architecture.

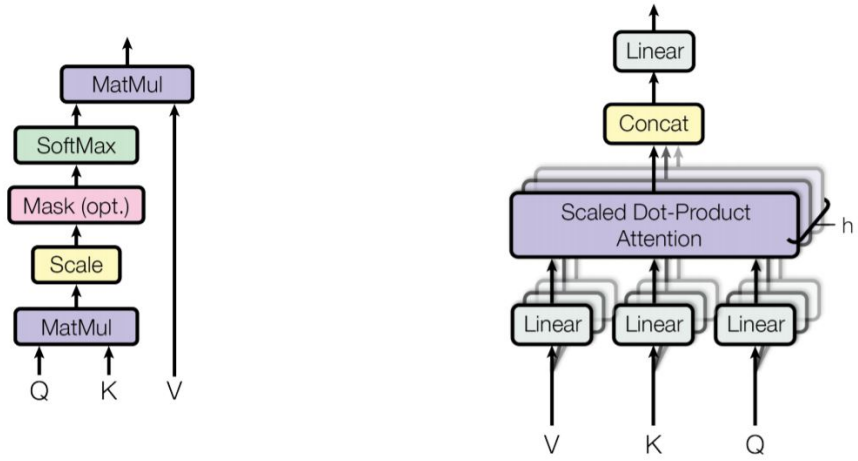

Figure 2.8.: (left)) Scaled Dot-Product Attention. (right) Multi-Head Attention; h denotes the number of attention layers running in parallel, from Vaswani et al. (2017)

Encoder The encoded word embeddings (with positional encoding) are used as the input to the encoder which consists of $N = 6$ layers each containing two sub-layers:

- a multi-head attention mechanism
- a position-wise fully connected feed-forward network

There are residual connection (He et al., 2016) employed around each of the two sub-layers to allow for a more effective gradient flow. Each sub-layer is subsequently normalized.

Decoder The decoder also consists of a stack of $N = 6$ identical layers. When looking at Transformer's architecture in Figure 2.7, we see that the decoder has an additional sub-layer performing multi-head attention over the output of the encoder stack. There are residual connections between each sub-layer, similarly to the encoder. The aim of the component "masked multi-head attention" is to only allow predictions for the position i which depend on the translated outputs at positions less than i.

Scaled Dot-Product Attention

Both the encoder and the decoder possesses a sub-layer called "multi-head attention". This component builds upon scaled dot-product attention, which operates on a query Q, key K and a value V. This attention function maps a query and a set of key-value pairs to an output. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function (depends on the attention mechanism) of the query with the corresponding key. Query, keys, values, and output are all vectors. Figure 2.8 (left) presents the scales dot-product attention:

$$Attention(Q,\ K,\ V) = softmax(\frac{QK^T}{\sqrt{d_k}})V, \tag{2.19}$$

where d_k is the dimension of the key. The scaling factor $\frac{1}{\sqrt{d_k}}$ improves computational speed and numerical stability (high values imply small gradients of the softmax function and, as a result, ineffective back-propagation of the error).

Multi-head attention mechanism

Vaswani et al. (2017) found it advantageous to linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. Multi-head attention mechanisms obtain h different representations of (Q, K, V), calculate scaled dot-product attention for each representation in parallel, yielding d_v-dimensional output values. As a next step, the results are concatenated and projected with a feed-forward layer. Figure 2.8 (right) presents multi-head attention mechanism.

$$MultiHead(Q,\ K,\ V) = Concat(\text{head}_1,\ ..., \text{head}_\text{h})W^O, \tag{2.20}$$

$$\text{where} \quad \text{head}_\text{i} = Attention(QW_i^Q\ ,\ KW_i^K,\ VW_i^V), \tag{2.21}$$

where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{mode}},\times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{mode}},\times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{mode}},\times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$. Matrices W_i and W^O are projection matrices the weights of which are learned. The benefit of multi-head attention is that the model can jointly attend to the information from different sub-spaces at different positions. Vaswani et al. (2017) extecuted their experiments with $h = 8$ parallel attention layers.

All experiments presented in Section 4.2 are executed using a Transformer architecture. A complete list of the parameters used to train the networks is presented in Section 4.7.

2.8. Source Factors

Sennrich and Haddow (2016) propose encoding linguistic features (e.g. morphological features, part-of-speech tags and syntactic dependency labels) into the neural network input in the form of source factors. They find that their incorporation provides further improvements in performance of MT models.

In our work, we explore incorporating NE information to signal NE occurrence using identical technique: *source factors*. As such, we wish to outline the way they are incorporated into the neural network input in more detail.

Source factors provide additional word-level information, are applied to the source language only, and take form of supplementary embeddings that are either added or concatenated to the word embeddings. This is illustrated with the following formula:

$$E \cdot x = \bigoplus_{f \in F} E_f \cdot x_{if} \tag{2.22}$$

where $\bigoplus \in \{\sum, \|\}$, $(\cdot)$ denotes a matrix-vector multiplication, E_f is a feature embedding matrix, x_i is the i-th word from the source sentence, and F is a finite, arbitrary set of word features. If source factors are added to the word embeddings, their size remains unchanged. In the concatenation event, the size of embedding matrix increases by the size of the concatenation vector.

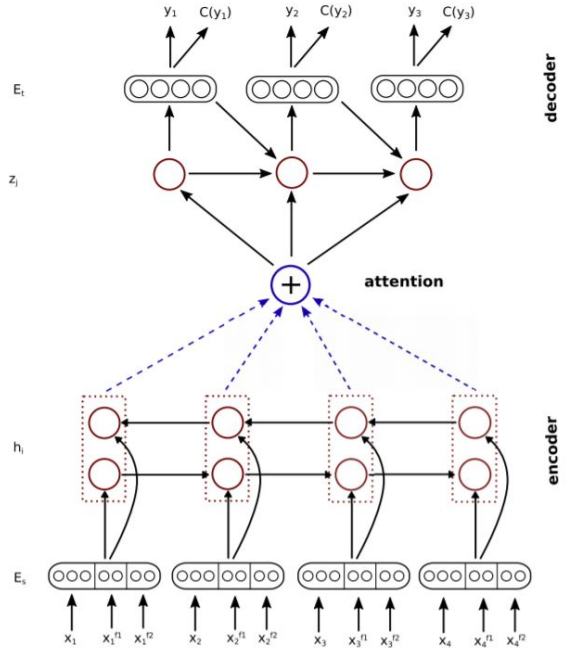

Figure 2.9.: An example of a NMT architecture allowing the integration of additional linguistic information as word features, directly into the input of the neural network, from Ha et al. (2017)

Figure 2.9 presents a NMT reference architecture where source factors are merged into the input. To each input token x_i an additional series of source factors $x_i^{f_j}$ is concatenated. When comparing the architecture from Figure 2.9 with the Equation 2.22, we state that the size of the feature set F equals to 3 (one regular input token and two word features).

2.9. Machine translation evaluation

As the popularity of MT grows, there is an increasing number of scientific groups researching new MT systems and neural architectures. The output of each of the new system must be evaluated and compared to the existing ones to be able to state whether it translates an identical input text better than previous approaches.

There are two possibilities of evaluating MT output: the manual and the automatic. The first one, also referred to as the human evaluation, is conducted by experts in translation and linguistics. This kind of evaluation is time-consuming and expensive. At the same time, it is inherently subjective, as on many occasions there does not exist only one fluent and accurate reference translation (Maučec and Donaj, 2019).

The alternative form of evaluation is the automatic approach. Its metrics are cost-free alternatives to the human evaluation. Furthermore, it is capable of estimating the improvement of a given MT system under development. The automatic evaluation relies on the availability of a human reference translation. It evaluates the output of MT systems by comparing it to the reference translation and aims to calculate some sort of similarity coefficient. As mentioned earlier, there is a great variability even in human translation.

Therefore, it is beneficial to provide more than one human reference translations for each of the machine translated sentence to be evaluated (Maučec and Donaj, 2019).

BLEU Evaluation Metric

There are several automatic evaluation metrics acknowledged by the MT community: BLEU (Papineni et al., 2002), NIST (Doddington, 2002), METEOR (Banerjee and Lavie, 2005), WER, TER (Maučec and Donaj, 2019) and others. In this work, we use BLEU as one of the evaluation metrics. As such, we present this metric in more detail.

The bilingual evaluation understudy (BLEU) score calculates similarity to a human translation and has been proposed by Papineni et al. (2002). More specifically, it counts the number of overlapping words between the hypothesis and reference translations. BLEU score ranges from 0-100. In general, the higher the score, the more accurate and fluent the translation is in comparison to the human translation.

As previously outlined, there may exist more than one appropriate reference translation. BLEU addresses this problem by allowing multiple reference translations for every source sentence. In order to achieve this, BLEU introduces the modified n-gram precision. An n-gram is a sequence of n subsequent words. BLEU measures the proportion of n-grams in the hypothesis (MT output) that also appear in the reference (human translation).

To address the problem of over-generation of frequent words, BLEU uses an alternative form of precision (modified precision). It only accepts as many occurrences of a word as actually appear in some reference text. A modified n-gram precision score p_n, already appropriately averaged over the entire test set $\mathfrak{T}$, is computed as follows:

$$p_n := \text{precision}_n^{\text{mod}}(\mathfrak{T}) = \frac{\sum\limits_{(H,\mathfrak{R})\in\mathfrak{T}} \sum\limits_{\text{n-gram}\in H} \text{count}_{\text{clipped}}(\text{n-gram}, H, \mathfrak{R})}{\sum\limits_{(H,\ \mathfrak{R})\in\mathfrak{T}} \sum\limits_{n-\text{gram}\in H} \text{count}(\text{n-gram},\ H)}, \tag{2.23}$$

where

$$\text{count}_{\text{clipped}}(\text{n-gram}, H, \mathfrak{R}) = \min(count(\text{n-gram}, H), \max_{R\in\mathfrak{R}}(count(\text{n-gram}, R))), \tag{2.24}$$

H denotes a hypothesis sentence, $count(\text{n-gram},\ H)$ is the number of occurrences of a specific n-gram in the hypothesis string H and $\mathfrak{R}$ is the set of references (Papineni et al., 2002). The aim of the term $count_{\text{clipped}}$ is to penalize the over-generation of words in the hypothesis. According to Papineni et al. (2002), shorter n-grams shall account for correct word choice (adequacy), longer n-grams for word order (fluency).

As precision favors shorter hypotheses, hypothesis sentences longer than their references are already punished by the modified n-gram precision measure. Consequently, a brevity penalty (BP) factor is introduced to penalize hypotheses that are too short. Let c be the length of the candidate sentence and r be the effective reference corpus length. The brevity penalty is computed as follows:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{2.25}$$

The final BLEU score is calculated as follows:

$$BlEU = BP \cdot exp(\sum_{n=1}^{N} w_n \log p_n) \tag{2.26}$$

where w_n are positive weights summing to one.

The BLEU score is widely acknowledged by the scientific community (Koehn, 2009). It is fast, simple to implement as well as language and domain independent. Furthermore, it does not necessitate additional resources nor adaptations required by a given specific task. Finally, it correlates well with the human judgment. Unfortunately, the BLEU score is not free from shortcomings. Certain researchers criticize BLEU for not being representative at the sentence level (scores rather have to be averaged over a large test set). Moreover, BLEU does not recognize similar phrasings or synonyms and, if present, scores them accordingly (Koehn, 2009).

3. Related Work

This chapter presents the related works focusing on the improvement of Named Entity (NE) translation in Machine Translation (MT). It starts by outlining early approaches to address this issue and continues by outlining transliteration methods. In Section 3.3 Deep Learning (DL) approaches to named entity translation are discussed.

3.1. Early approaches to named entity translation

The problem of named entity translation has been studied for a long time. Conventionally, dictionaries have been found useful for a dictionary-based machine translation. The weakness of this approach emerges in the scenario when a specific term is not included in the dictionary list. The "out-of-dictionary" terms concern names, such as people, places, companies and products especially (Dale et al., 2000).

Newmark (1981) observes that in human translation certain type of named entities are left untranslated and are copied into the target language. Based on this argument, Babych and Hartley (2003) propose creating "do-not-translate lists" with named entities, with special focus on the organizations' names, which should be kept verbatim in the translated text. For this purpose, they use a Named Entity Recognition (NER) system to recognize named entities in the text and create the aforementioned lists. They achieve significant performance improvements and conclude that "combining present-day MT systems with specific IE modules has a beneficial effect on the overall MT quality".

3.2. Transliteration approaches to named entity translation

Depending on the type of a named entity, MT systems decide if they should be meaning-translated or phoneme-transliterated (Al-Onaizan and Knight, 2002). Typically this relies upon the type of the named entity. For instance, personal names tend to be transliterated. Transliteration is concerned with replacing words in the source language with their approximate phonetic or spelling equivalents in the target language (Shaalan, 2014). It originated in the early 2000s as part of machine translation to deal with proper nouns and technical terms that are translated with preserved pronunciation (Karimi et al., 2011).

Statistical Machine Translation (SMT)

Knight and Graehl (1998) research this topic with the transliteration approach using cascaded probabilistic finite-state transducers. In their work, they implement transformation rules for back-transliteration from Japanese to English. For this purpose, they build a

weighted finite-state automaton modeling the empirically estimated probability distributions for back-transliterating English words written in katakana into their original English form.

Haizhou et al. (2004) indicate the limitations of the previous approach and state that phoneme-based approaches must be more language-specific as different languages have varying phonic rules. They propose a framework allowing a direct orthographical mapping between two different languages without intermediate phonemic representation.

Motivated by the out-of-vocabulary problem, occurring especially when transcribing named entities, Freitag and Khadivi (2007) develop a a technique which combines conventional MT methods with a single layer perceptron. They present a discriminatively trained sequence alignment which exploits arbitrary features of the input strings.

Hermjakob et al. (2008) investigate the problem of when transliteration is the most promising and integrate named entity translation into traditional MT systems. In detail, they use a tagger to identify good candidates for transliteration and add transliterations to the Statistical Machine Translation (SMT) phrase table dynamically. During decoding time these new translation candidates can directly compete with translations during decoding.

Rama and Gali (2009) view the process of transliteration as a process of translation at the character level, without re-ordering. From this perspective, it is possible to directly apply a phrase-based SMT system to address the task of transliteration. In their work, they evaluate several techniques for sequence-pair extraction for transliteration.

Neural Machine Translation (NMT)

Deselaers et al. (2009) use deep belief networks for machine transliteration. Their work did not deliver competitive results (at the time), yet is demonstrated one of the early applications of deep belief networks to transliterate named entities. In detail, no finite-state machines or phrase-based techniques are used. The system is not dependent on word alignments and beam-search decoding. Furthermore, their analysis includes an evaluation of the optimal network structure and size as well as demonstrates reordering capabilities and the creation of multiple hypotheses.

Kundu et al. (2018) experiment with different deep learning architectures for machine transliteration. Specifically, they train a encoder-decoder RNN and a convolutional sequence-to-sequence (Conv Seq2Seq) network. This is the first attempt to use a convolutional Seq2Seq approach in transliteration of named entities. In their work, they also present a novel ensemble method based on counting the frequencies of hypotheses and creating a voting classifier based on the output. Their framework has been submitted to the 2018 NEWS Shared Task on Transliteration and achieved top performance for the En–Pe and Pe–En language pairs and comparable results for other tracks.

Finally, Grundkiewicz and Heafield (2018) present a modern day approach to NE transliteration. They train a deep attentional RNN encoder-decoder network and apply state-of-the-art techniques from NMT, such as dropout regularization (prevents overfitting), model ensembling, rescoring with right-to-left models, and back-translation. Their submission to the NEWS 2018 Shared Task on Named Entity Transliteration achieved best results in several tracks.

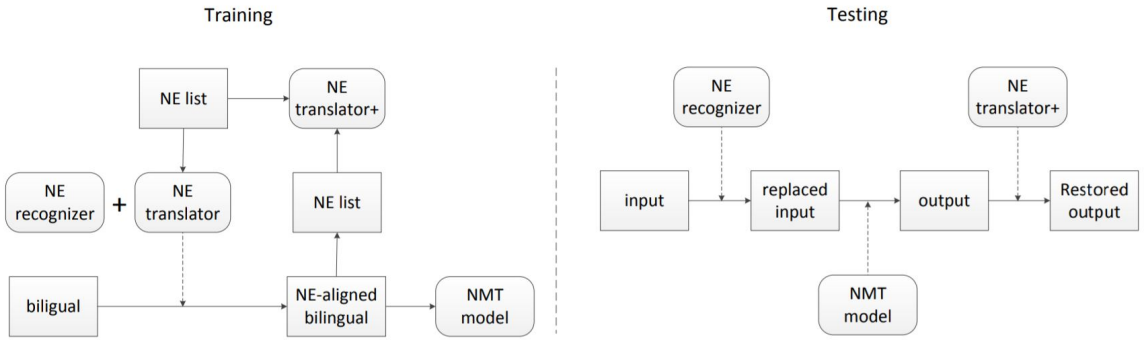

Figure 3.1.: System architecture incorporating an external named entity translation component, from Li et al. (2018a)

3.3. Deep Learning approaches to named entity translation

Neural Machine Translation (NMT) has recently shown promising results, replacing Statistical Machine Translation (SMT) as state-of-the-art approach to machine translation. Technological advances, such as sequence-to-sequence (Sutskever et al., 2014), attention mechanism (Luong et al., 2015) and Transformer networks (Vaswani et al., 2017) greatly contributed to improving accuracy and fluency of machine translation.

Several research groups propose translating named entities prior to the translation of the whole sentence by an external named entity translation model. Li et al. (2018a), Yan et al. (2018), and Wang et al. (2017) follow the “tag-replace” training method using an external MT model. Figure 3.1 presents their architecture in detail. During the training phase, first a character-level sequence-to-sequence model is trained on an external named entity list (e.g. from Wikipedia titles and Chinese-English named entity lists v1.0 in this particular case). This list may be adjusted according to the category information. As a next step, such trained NE translator and NE recognizer are used to create aligned NE pairs based on the training corpus for the task at hand. This step results in a list of NE pairs which is, subsequently, combined with the externally trained NE translator to further improve its performance. Similar to Luong et al. (2014), the aligned NE pairs are then replaced with their NE class symbols, resulting in sentence pairs as shown in Figure 3.2. Finally, such tagged data is used to train the MT model.

ZH: LOC1 重新 开放 驻 LOC2 大使馆

EN: LOC1 reopens embassy in LOC2

Figure 3.2.: Tag-replace training method, from Li et al. (2018a)

During inference, all named entities in the source sentence are replaced by a respective tag. After the model has translated the sentence, a post-processing step is employed to recover the translation of the replaced NEs. In this scenario, the NE translation module is used. While these works achieve successful results in named entity translation, the drawbacks of the "tag-replace" method include the necessity to train a second translation

model, alignment errors and the loss of context information while translating named entities, which e.g. impedes morphological adaptation.

Li et al. (2018b) explores a different approach, namely inserting inline annotations into the data providing information about named entity features. Such annotations are inserted into the source sentence in form of XML tags, consisting of XML boundary tags and NE class labels. Figure 3.3 presents the inline annotation technique of Li et al. (2018b).

Original Source:
`Patrick Roy resigns as Avalanche coach`
Words and subwords with NE tags[3]*:*
`<PERSON> Patrick Roy </PERSON> resigns <ORG> Avalan @@che </ORG> coach`

Figure 3.3.: Inline annotation applied to a source sentence after tokenization and sub-words splitting, from Li et al. (2018b)

Section 4.1.3 presents the approach of Li et al. (2018b) in more detail. Both of these approaches (the inline annotation approach and the tag-replace approach) do not modify the original sequence-to-sequence NMT architecture and the network can learn by just an augmentation of the training data.

Recently, researchers have shown the benefit of explicitly encoding linguistic features, in form of source factors (described in Section 2.8), into NMT (Sennrich and Haddow, 2016; García-Martínez et al., 2016). Sennrich and Haddow (2016) find that adding morphological features, part-of-speech tags, and syntactic dependency labels as input features improves translation quality. Their main innovation over the standard encoder-decoder architecture is the ability to represent encoder input as a combination of features (source factors) which are subsequently concatenated or added to the embedding vector. Dinu et al. (2019) use source factors successfully to enforce terminology.

The use of encoding linguistic features in form of source factors to improve named entity translation has been performed by Ugawa et al. (2018). The work of Ugawa et al. (2018) is similar to ours, in the way that they also incorporate NE tags with the use of source factors into the NMT model to improve named entity translation. They, however, introduce an additional chunk-level long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997b) layer over a word-level LSTM layer into the encoder to better handle compound named entities. Additionally, they use a different network architecture (LSTM), and apply a different annotation technique (IO tagging) than we explore (IOB tagging).

Furthermore, the work of Ugawa et al. (2018) lacks a fair assessment of the quality of named entity translation. In detail, they based their conclusions on inconsistent BLEU improvements across three language pairs. Hermjakob et al. (2008) state that "general MT metrics such as BLEU, TER, METEOR are not suitable for evaluating named entity translation and transliteration, because they are not focused on named entities". In this thesis, we provide an extensive evaluation of the NE quality translation (Section 6.3), including a human assessment (Section 6.4).

Our approach is outlined in full detail in Sections 4.1 and 4.2.

4. System Description

This chapter starts by outlining the research questions which this thesis aims at resolving. Further, it describes the conducted experiments and outlines the language pairs based on which they are executed. As a next step, Section 4.4 illustrates data pre-processing with regard to tokenization and data formatting.

The chapter later continues with the analysis of the properties of the training data sets. Moreover, it present two algorithms: Generate Source Factors and Generate Inline Annotations both of which are used to annotate named entities in the data. Finally, the functionality of the Sockeye NMT toolkit is presented together with its configurations and training parameters which are used to execute the experiments.

4.1. Research questions

This thesis contributes to the area of named entity translation. We explore incorporating named entity information as additional parallel streams with the use of source factors (described in Section 2.8) to signal named entity occurrence. In the upcoming examples source factors are represented as indices. Our experiments focus on annotating named entities in the source sentence only. No annotation is done on the target side. This section outlines the research questions and presents studied annotation techniques.

4.1.1. Influence of the granularity of named entity classes

The thesis aims at examining whether the differentiation between different types of named entities is beneficial to the NMT systems, and whether narrower and more detailed classes could produce better NE translation. In other words, we explore whether the NE class granularity may influence translation quality and help decrease word ambiguity. To assess it, we define two cases:

- **a "fine-grained" case** – Here we use specific NE class labels (e.g. person, location, organization) to denote named entities in the source sentence. Value (0) is used for a regular *sub-word* (default), (1) for NE class *Person*, (2) for NE class *Location*, (3) for NE class *Organization*.
- **a "coarse-grained" case** – Here we use two source factor values only: (0) as default and (1) to denote a named entity in a generic manner.

We annotate source sentences with an external NER system. Examples for the different annotation strategies (that we experiment with) are presented in Table 4.1. Each sub-word is assigned an index denoting its corresponding source factor value[1].

[1] The way how source factors are being incorporated into the neural network input is described in Section 2.8.

Language	Variant	Sentence
English	BPE only	Belfast - Gi@@ ants won thanks to Patri@@ ck D@@ w@@ yer
English	fine-grained	Belfast_{2} -_{0} Gi@@_{3} ants_{3} won_{0} thanks_{0} to_{0} Patri@@_{1} ck_{1} D@@_{1} w@@_{1} yer_{1}
English	coarse-grained	Belfast_{1} -_{0} Gi@@_{1} ants_{1} won_{0} thanks_{0} to_{0} Patri@@_{1} ck_{1} D@@_{1} w@@_{1} yer_{1}

Table 4.1.: Annotation to assess the influence of named entity labels' granularity; i. fine-grained: (0) for a regular *sub-word* (default), (1) for NE class *Person*, (2) for NE class *Location*, (3) for NE class *Organization* ii. coarse-grained: (0) default, (1) to denote a NE

In the "fine-grained" case: `Belfast` is classified by an NER system as location, therefore its source factor value is (2), `Giants` as an organization and receives the value (3) and finally `Patrick Dwyer` is recognized as a person and is assigned the value (1). In the coarse-grained case all named entities are marked with value (1).

A model trained with no external annotation is called a baseline. We refer to the models which are trained with source factors as "annotated" models. Based on this nomenclature, we define the following research questions:

Research Question 1 *Do annotated models achieve a better named entity translation rate in comparison to the baseline? Is named entity annotation in the form of source factors helpful?*

Research Question 2 *Is there a difference between fine-grained and coarse-grained annotation? Do specific named entity class labels contribute to an improved named entity translation?*

4.1.2. Inside-outside-beginning (IOB) tagging

Additionally, we investigate whether inside–outside–beginning (IOB) tagging (Ramshaw and Marcus, 1999) used to signalize where a NE begins and ends as a second input feature may guide models to translate compound named entities better. In IOB tagging, (B) indicates the beginning, (I) the inside and (O) the outside of a NE (a regular word). Thanks to the IOB tagging, the network receives a signal which words belong together, after having been split by the BPE algorithm (described in Section 2.4.6). The same logic applies to compound named entities. Table 4.2 displays an example of the IOB annotation method.

Language	Variant	Sentence
English	IOB tagging	$\text{Belfast}_{\text{B}}$ -_{O} Gi@@_{B} ants_{I} won_{O} thanks_{O} to_{O} $\text{Patri@@}_{\text{B}}$ ck_{I} D@@_{I} w@@_{I} yer_{I}

Table 4.2.: IOB annotation denoting compound named entities; (B) indicates the beginning, (I) the inside and (O) the outside of a NE (a regular word)

Research Question 3 *Is IOB tagging helpful to translate compound named entities?*

4.1.3. Inline Annotation

As our goal resembles that of Li et al. (2018b) (their work is described in Section 3.3), we compare approaches from Sections 4.1.1 and 4.1.2 against their inline model annotation method with XML boundary tags. Inline annotations are inserted directly into the source sentence. No additional input streams are provided. Each named entity is encompassed with an XML boundary tag denoting the start and end of this named entity. The XML tag content marks the named entity class.

Li et al. (2018b) use specific NE class labels, which correspond to the "fine-grained" case in our work. We refer to their approach as "Inline Ann. (fine-grained)" and present this annotation method in Table 4.3.

Language	Variant	Sentence
English	Inline Ann. (fine-grained)	<LOC> Belfast </LOC> - <ORG> Gi@@ ants </ORG> won thanks to <PER> Patri@@ ck D@@ w@@ yer </PER>

Table 4.3.: Inline annotation: XML markup shows the begin and the end of each named entity

Research Question 4 *How does inline annotation perform in comparison to named entity annotation with source factors?*

4.1.4. Source factors combination methods

As outlined in Section 2.8 linguistic features (source factors) can be either added or concatenated to the word embeddings. We would like to investigate whether one method is superior to the other. As such, we formulate the question which vector combination method performs better.

Research Question 5 *Does vector concatenation perform better than vector addition? Can a clear recommendation be defined for any given language pair?*

4.2. Experiments

Based on the research question presented in Section 4.1, we define the following experiments:

Model no.	Label type	Variant	IOB
1.	fine-grained	sum	no
2.	fine-grained	concat 8	yes
3.	fine-grained	sum	yes
4.	coarse-grained	concat 8	yes
5.	coarse-grained	sum	yes
6.	Inline Ann. (fine-grained)		no
B	Baseline		no

Table 4.4.: Experimental setup – named entity annotation configurations

Table 4.4 displays the experiments. Column "Model no." provides the model id; column "Label type" denotes whether specific ("fine-grained") or generic ("coarse-grained") NE labels are used; column "Variant" describes whether source factors are added ("sum") or concatenated ("concat") to the word embeddings; column "IOB" describes whether IOB tagging is used as a second source factor stream. We use the source factor embedding of size 8 for the concatenation case ("concat 8").

Each of the planned experiments is designed to help in answering one or more research questions from Section 4.1. The following list explains how we intend to address them:

- Research Question 1 – Compare the performance of models (1-5) with the baseline (B).
- Research Question 2 – Compare the performance of models (2) and (3) with models (4) and (5).
- Research Question 3 – Compare the performance of model (1) and with models (2-5) and the baseline (B).
- Research Question 4 – Compare the performance of models (6) and with the rest of the models and the baseline (B).
- Research Question 5 – Compare the performance of models (2-5).

Terms, such as "model performance" or "model comparison", are explained in Chapter 6. Sections 6.3.2 and 6.4.3 provide scientific findings based on which we answer the research questions.

4.3. Language pairs and training data

This section outlines the language pairs as well as the training and validation data used to train the models from Section 4.2.

4.3.1. Training and validation data

The findings of this master thesis will be based on the following language pairs:

- English-German (En-De)
- English-Chinese (En-Zh)

We train NMT systems for English→German and English→Chinese on the news translation task data provided for the fourth Conference on Machine Translation (WMT2019). All corpora are available on the official WMT2019 website[2] and are merged into a parallel training data set.

Language pair: En-De For the translation from English to German the following data sets are chosen:

- europarl (v9), Koehn (2005) – The Europarl parallel corpus is extracted from the proceedings of the European Parliament. It includes versions in approx. 21 European languages.
- news-commentary (v14), Tiedemann (2012) – The corpus was created as training data resource for the First Conference on Machine Translation (WMT2016) and consists of political and economic commentary crawled from the web site Project Syndicate.[3]

Language pair: En-Zh For the translation from English to Chinese the following data sets are chosen:

- news-commentary (v14), Tiedemann (2012) – (as described above)
- United Nations Parallel Corpus v1.0, Ziemski et al. (2016) – The UN corpus is composed of the official records and other parliamentary documents of the United Nations manually translated between 1990 and 2014, and offers six official languages of the UNs.

  In this work, this data set is shortened to match the size of the training data set for English→German by using the newest data from the end of the corpus for training, see also Table 4.5.

Validation/Development data We use *newstest2018* validation data from the WMT2019 news translation challenge. This validation data was used as a test set for the challenge from the previous year and has been proved by human translators.

[2] `http://www.statmt.org/wmt19/translation-task.html#download`
[3] `https://www.project-syndicate.org/`

4.3.2. Training data statistics

It is imperative to ensure that training data has enough named entities which are recognized by the spaCy NER system[4]. Should there be not sufficient named entities in the training data set, the NMT models may not be able to learn the semantics behind the source factors.

Data suitability is evaluated by counting the number of named entities and setting their amount in relation to the overall number of tokens and to the number of sentences in the corpus.

Metrics	En-De	En-Zh
Number of tokens in corpus	58,877,917	51,230,838
Number of NE	1,846,171	2,333,623
Relation: NE to all words	≈ 3.14 %	≈ 4.55%
Number of sentences	2,146,644	2,128,234
Number of sentences without NE	744,758	560,722
Number of sentences with other NE classes[5]	319,013	423,967
Number of sentences with NE	1,082,873	1,153,545
Percentage of sentences with NE	≈ 50.44%	≈ 53.95%

Table 4.5.: Number of named entities in WMT2019 training data sets: En-De and En-Zh

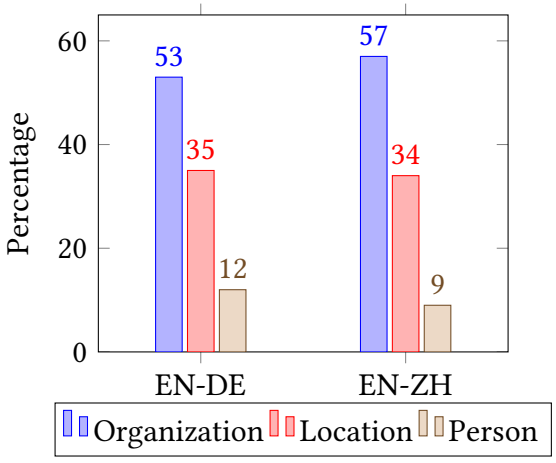

Figure 4.1.: Categorization of named entities in training data sets: En-De and En-Zh, in %

Table 4.5 displays the number of named entities occurring in the training data as recognized by spaCy NER. There is an almost equal percentage of sentences in both corpora which contain named entities, 50.44% and 53.95% for En-De and En-Zh respectively, as well as similar number of overall sentences, approx. 2.1 million. The same findings hold true for validation/development data (its statistics are available in Appendix A.1).

Figure 4.1 displays the percentage categorization of available named entities among three classes: Organization, Location and Person. There occurs almost identical division

[4]The selection process of this NER system is presented in Chapter 5.

[5]A full list of NE classes recognized by spaCy NER is outlined in Appendix A.2.

among named entity classes across three categories (around 55% for NE label *Organization*, 35% for NE label *Location*, 10% for NE label *Person*). As such, the properties of training data for both language pairs are very similar.

4.4. Data pre-processing

This section describes the pre-processing steps applied to the training data from Section 4.3. The pre-processing pipeline applied to the raw training data aims at bringing it into the format conventionally used to train NMT models. Sections 4.4.1 and 4.4.2 present the detailed pre-processing steps applied to the training data for the respective language pair (during the inference time identical pre-processing steps are applied). Data pre-processed in this way is used to train the experiments described in Section 4.2.

4.4.1. English-German

This subsection presents the exact tools used to pre-process the training corpus for translating from English to German.

- **Tokenization** – The source and the target side of the corpus are tokenized using the Spacy Tokenizer[6]. It is a rather simplistic rule-based tokenizer. First, it splits the sentence on whitespace characters, then it performs two checks on each token by going through them from left to right:
  1. Check if a substring matches an exception rule, e.g. `don't` does not contain whitespace, but should be split into two tokens, `do` and `n't`, while `U.K.` should always remain one token.
  2. Check if prefix, suffix or infix should be split off on punctuation marks like commas, periods, hyphens or quotes, e.g. `three-year-old`, is split to 5 tokens (= at each hyphen).

  A detailed description of the way Spacy Tokenizer functions can be found on the spaCy official website[7]. This tokenizer was chosen to match the Named Entity Recognition system selected in Chapter 5.

- **Byte-Pair Encoding (BPE)** – (described in Section 2.4.6) A joint source and target BPE encoding was performed using 32k merge operations. We used the implementation[8] developed by Sennrich et al. (2016).

4.4.2. English-Chinese

This subsection presents the exact tools used to pre-process the training corpus for translating from English to Chinese.

First, we start by outlining the tools for the English side of the corpus:

[6] `https://spacy.io/api/tokenizer`
[7] `https://spacy.io/usage/linguistic-features#how-tokenizer-works`
[8] `https://github.com/rsennrich/subword-nmt`

- **Tokenization** – The English side of the corpus is tokenized using the spaCy Tokenizer.
- **Byte-Pair Encoding (BPE)** – An independent BPE encoding was learned and applied using 32k merge operations.

Two possibilities to pre-process the Chinese side of the training corpus are considered:

- **word-level Chinese**, as described in Bawden et al. (2019) – This approach uses a simple tokenization tool[9] (mode "conservative") which splits only at punctuation marks, and leaves continuous Chinese character strings untouched). On top of the tokenized data set, BPE (described in Section 2.4.6) is applied.
- **character-level Chinese** – According to Emerson (2005) a Chinese word contains on average 1.6 characters. Bawden et al. (2019) states that using raw Chinese characters in tokenized text makes sense as they form natural subword units (= words). Their approach segments all Chinese sentences into characters, but keeps non-Chinese signs unsegmented in order to allow for English words and numbers to be kept together as individual units. The segmentation strategy of Bawden et al. (2019) has been found successful for translating into Chinese, however produces significantly worse results when translating from Chinese into English.

Finally, the first (word-level Chinese) approach was chosen, using OpenNMT (mode conservative) as tokenizer. As a next step, an independent BPE encoding was learned and applied using 32k merge operations.

4.5. Data annotation

The previous section (4.4) outlined how the training corpora are pre-processed to be consumed by the NMT toolkit. This section focuses on describing the process of named entity annotation in the training data (at inference time identical pre-processing is applied as well). It describes the mechanisms to generate source factors (Sections 4.5.2 and 4.5.3) as well as to insert inline annotations (XML markup) (Section 4.5.4) into source sentences.

4.5.1. Process Overview

This subsection provides preliminary information about the annotated models (trained with source factors).

Annotated Models The annotated models (described in Subsection 4.1.1) and models with IOB tagging (described in Subsection 4.1.2) require named entity annotation. We use spaCy Named Entity Recognition (NER) system[10] to recognize and classify named entities in the source sentences. For every source sentence in the training data (after applying BPE), we generate two lines with source factors:

[9] `https://github.com/OpenNMT/Tokenizer`
[10] The selection process of this NER system is presented in Chapter 5.

(i) marking named entities (either the coarse-grained or the fine-grained case),

(ii) marking IOB tagging.

The baseline model is trained with no external annotation. This implies that there are no additional source factor files generated to train the baseline.

4.5.2. Annotation of named entity classes with source factors

This subsection describes the process of source factor generation in more detail. It focuses on outlining the algorithm which generates source factor text files which are later one of the input sources for the NMT toolkit.

The aim of the Algorithm Generate Source Factors (Algorithm 1) is to assign a source factor value to each of the sub-words in any given sentence. The source factor value depends on whether the fine-grained or coarse-grained case is processed.

Algorithm 1 Generate Source Factors

For every source sentence *i* (in the BPE encoded form) in the corpus, we do the following:

1. We create a duplicate of the sentence *i* and remove all signs '@@ ' from it. By doing so, we receive the original sentence (still tokenized) from the corpus and call it *o* (which stands for the "original sentence"). Let us illustrate this based on the following example:

Sentence *i*:	Belfast - Gi@@ ants won thanks to Patri@@ ck D@@ w@@ yer .
Sentence *o*:	Belfast - Giants won thanks to Patrick Dwyer .

The generation of sentence *o* from sentence *i*

2. We annotate sentence *o* with spaCy NER. SpaCy provides the start and end positions of each named entity. We generate a source factor string according to the words in the sentence *o* and named entity positions (recognized by spaCy's). When doing it, we differentiate between the fine-grained and coarse-grained cases.

Sentence *o*:	Belfast	-	Giants	won	thanks	to	Patrick	Dwyer
Fine-grained	2	0	3	0	0	0	1	1
Coarse-grained	1	0	1	0	0	0	1	1

The annotation of each word with its source factor value

Thanks to step 2, we have an assignment between each word in sentence *o* to its source factor value. Now, we translate this assignment to match the length of sentence *i*, which is in BPE form.

Algorithm 1 Generate Source Factors - cont.

3. This step describes the extension of the source factors string to the sub-word level. We loop over each sub-word from sentence i (in BPE form). We recognize words which were split by the BPE algorithm as they contain the sign `@@` and count the number of sub-words each word was split into.

   Let n denote the number of sub-words the word w was split into. For each word w, we insert a source factor value (which we know from previous step) n times into the source factor string.

   For example, word `Giants` was split into two sub-words ($n = 2$). Therefore, value (3) is inserted into the final source factor value two times (or value (1) if coarse-grained case is processed).

Belfast	-	Gi@@	ants	won	thanks	to	Patri@@	ck	D@@	w@@	yer
2	0	3	3	0	0	0	1	1	1	1	1
1	0	1	1	0	0	0	1	1	1	1	1

Insertion n number of times of the corresponding source factor value

4. Finally, we continue with sentence $i + 1$. Each source factor string is stored in a separate text file in the same order as sentences appear in the training corpus. This format is required by Sockeye, our NMT toolkit (described in Section 4.6). Values are split by a single space sign.

We generate the source factors, based on the pre-processed data. The fact that data is pre-processed implies that sentences are tokenized and that the BPE algorithm has already been applied. BPE splits words into sub-words and uses the sign `@@` to mark the position where the division took place[11].

4.5.3. Annotation of named entity boundaries with source factors

Research question 3 intends to determine whether Inside-Outside-Beginning (IOB) tagging helps the networks to recognize compound named entities and, as a result, translate them with less errors.

The algorithm Generate IOB Tagging which generates IOB tags is similar to Algorithm 1. The difference lies in Step 3. Here, instead of inserting the source factor value denoting the named entity class, either (B) or (I) are inserted depending on the number of sub-words the word w was split into. If the sub-words of word w contain the sign `@@`, (B) and (I) will be used (as in `Giants` case). If word w was not split ($n = 1$), then only (B) is inserted (as in `Belfast` case).

[11] This symbol is used as a separator in the implementation by Sennrich et al. (2016).

Belfast	-	Gi@@	ants	won	thanks	to	Patri@@	ck	D@@	w@@	yer
B	O	B	I	O	O	O	B	I	I	I	I

The generation of (B), (I) and (O) tags according to the position of sub-words in the sentence

4.5.4. Inline Annotation with XML markup

Research question 4 intends to determine whether inline annotation (described in Section 4.1.3) performs better than the annotated models.

The aim of the Algorithm Generate Inline Annotations (Algorithm 2) is to insert inline annotations into every source sentence in the corpus.

Algorithm 2 Generate Inline Annotations

For every source sentence i (in the BPE form) in the corpus, we do the following:

1. We remove all signs `@@` from sentence i and create, thus, sentence o (the original sentence from the corpus).
2. We loop over each sub-word in sentence i and create a function f mapping words from the original sentence o onto a list of its sub-words generated by the BPE algorithm. Function f has as many arguments as there are words in the original sentence.

   For example, the following mappings are created:

$$
\begin{aligned}
f(Belfast) &= ['Belfast'] \\
f(Giants) &= ['Gi@@', \quad 'ants'] \\
f(thanks) &= ['thanks'] \\
f(Dwyer) &= ['D@@', \quad 'w@@', \quad 'yer']
\end{aligned}
$$

3. We run spaCy NER over sentence o and receive a list of recognized and classified named entities in this sentence together with their positions.
4. We insert XML boundary tags according to the positions from previous step into sentence o.
5. We replace each word in sentence o with its BPE sub-words with the use of function f.

4.6. Sockeye: The NMT toolkit

This section presents the NMT toolkit which is used to conduct the experiments from Section 4.2.

We use the Sockeye machine translation framework (Hieber et al., 2017) for our experiments[12]. Sockeye is a sequence-to-sequence framework for Neural Machine Translation based on Apache MXNet Incubating[13]. It implements the well-known encoder-decoder architecture with attention.

Sockeye supports a number of up-to-date features, such as:

- multiple model architectures: Encoder-Decoder (Wu et al., 2016), Convolutional (Gehring et al., 2017a), Transformer (Vaswani et al., 2017)
- multiple supported attention mechanisms (dot, mlp, bilinear, multihead-dot, encoder last state, location)
- cross-entropy label smoothing, e.g. Pereyra et al. (2017)
- layer normalization (Ba et al., 2016)

The Sockeye toolkit is free software released under the Apache 2.0 license.

4.7. NMT architecture

This section describes the NMT architecture used to execute the experiments from Section 4.2.

We train our models with a Transformer (Base) network (Vaswani et al., 2017) architecture[14]. While we use a state-of-the-art encoder-decoder Transformer network, our approach does not modify the standard NMT model architecture, thus can be applied to any sequence-to-sequence NMT model. Tables 4.6 and 4.7 present training configurations and training parameters. Each model is trained with configurations presented in these tables.

Parameter name	Value
encoder type	transformer
decoder type	transformer
no. of encoding layers	6
no. of decoding layers	6
no. of hidden units in transformer layer	512
no. of heads for all self-attention	8
no. of hidden units in transformers feed forward layers	2048
activation function in feed forward layers	relu

Table 4.6.: The architecture of the Transformer network used across all experiments

[12] `https://awslabs.github.io/sockeye/index.html`
[13] `http://mxnet.incubator.apache.org/`
[14] A detailed description of the Transformer architecture is outlined in Section 2.7.

Parameter name	Value
optimizer for SGD update rule	Adam
batch size	4096 tokens
word embedding size	512
evaluation metric on validation data	perplexity
dropout probability for multi-head attention	0.1
maximum sequence length	100 tokens
source factor embedding size[15]	8
random seed	13

Table 4.7.: NMT Sockeye training parameters (used across all experiments)

Experimental setup Each model is trained on 1 GPU Tesla T4. Training finishes if there is no improvement for 32 consecutive checkpoints on the validation data *newstest2018* (validation data from the WMT2019 news translation task).

Baseline and Inline Annotation models The baseline model and the Inline Annotation models (model 6 in Table 4.4) are trained with no source factors.

Annotated models The annotated models are trained with some additional parameters with which Sockeye `train` commands are started. Table 4.8 presents their values.

Parameter name	Value
source-factors-combine	sum/concat
source-factors-num-embed	8

Table 4.8.: NMT Sockeye training parameters for the annotated models

[15] only for the concatenation case

5. Selection of the Named Entity Recognition system

This chapter focuses primarily on outlining the decision process taking place to determine which Named Entity Recognition (NER) system to use for annotating named entities to execute the experiments from Section 4.2. Tested systems are investigated by examining their recognition quality (Section 5.3) and annotation performance (Section 5.4). The aim of these analyses is to assess the suitability of NER system candidates and to choose the best available system. Additionally, this chapter provides a list of researched named entity classes which are annotated in the training data and at inference time, outlines reasons for their selection and provides their examples (Section 5.2).

The language chosen for the analyses in this section is determined as follows: According to the research questions (Section 4.1) and the experiments' list (Section 4.2) named entities on the source side of the corpus are annotated. As stated in Section 4.3, English is the source language for all executed experiments. As such, English models of NER system candidates are examined.

5.1. NER system candidates

The following three NER systems have been considered for the work at hand. They seem to be commonly used within the NLP community. Table 5.1 outlines the training data used to train their pre-trained models.

- **spaCy NER**[1] – is an open-source library for advanced Natural Language Processing in Python. The spaCy NER system is Deep Learning-based and contains a word embedding strategy using sub-word features and "Bloom" embeddings, a deep convolutional neural network with residual connections, and a novel transition-based approach to named entity parsing[2]. The default model identifies a variety of named and numeric entities.

  spaCy NER provides three pre-trained English models which differ in model size and F1-scores. We decided to select the middle-sized model (`en_core_web_md`) for the following reasons. First, the large model is almost 9 times larger and has, therefore, longer loading times. Secondly, according to the official spaCy's documentation, both models have almost identical accuracy on the same test set (around 85%)[3]. The name of the test set is not specified in the documentation.

[1] `https://spacy.io/usage/linguistic-features#named-entities`
[2] `https://spacy.io/universe/project/video-spacys-ner-model`
[3] `https://spacy.io/models/en`

- **NLTK NER**[4] (Natural Language Toolkit) – NLTK is a leading platform for building Python programs to work with human language data. It contains a feature-based NER system. The Chunker and the NER modules are trained on the ACE corpus with a Maximum Entropy model (Loper and Bird, 2002).
- **Stanford NER**[5] (Finkel et al., 2005) – is a feature-based Named Entity Recognizer implemented in Java. The software provides a general implementation of (arbitrary order) linear chain Conditional Random Fields (CRF) sequence models. The system is trained on local word features such as the spellings of the current and surrounding words. In addition, distributional similarity features are included, which provide information on where the considered word tends to appear in the corpus (Manning et al., 2014). The Viterbi algorithm (Forney, 1973) is used for decoding, e.g. to determine the best sequence of tags.

  The English model with 3 classes is selected. It is trained on the largest amount of training data and has, therefore, the highest potential for good performance.

The above NER candidates are investigated by examining their quality of recognition (Section 5.3) and annotation performance (Section 5.4).

5.2. Researched Named Entity Classes

This section focuses on outlining the researched named entity classes and provides reasons for their choice.

As NE Recognition is an active research field and the search for the best recognition methods continues, the quality of NER systems may vary under different research scenarios and domains (Goyal et al., 2018). Incorrect NE annotation in the data may influence the results of this work negatively. Table 5.1 presents NE classes recognized by studied NER systems.

NER system	Recognized classes	Trained on
spaCy NER	Person, Organization, Location and others[6]	OntoNotes5, Common Crawl
NLTK NER	Person, Organization, Location, Geo-Political Entity, Date, Time, Money, Percent, Facility (Infrastructure)	ACE 2004 Multilingual Training Corpus[7] (numerous corpora and trained models available[8])
Stanford NER	Location, Person, Organization (model with 3 classes)	CoNLL-2003, MUC-6, MUC-7, ACE 2002 Multilingual Training Corpus

Table 5.1.: List of supported NE classes by examined NER systems

[4] `https://www.nltk.org/book/ch07.html` (Section 5)
[5] `https://nlp.stanford.edu/software/CRF-NER.html`

Each NER system is trained using different training datasets and may have different definitions of the same NE class. It may prove to be challenging to unify the definitions and ensure a fair evaluation of multiple NER systems.

With the above observations in mind, we focus on three most established and well-researched NE classes. These are: *Person*, *Location* and *Organization*. We limit, thus, the possibility of incorrect annotation and inaccurate evaluation. Examples of different named entity instances are presented below:

- *Person* – Kim Jong-il, Robert Fogel, Reagan, Abu Ghraib, Sarkozy, Patrick Dwyer, Franziskus Beauvillier, Walid al-Moualem
- *Organization* – The United Nations, World Bank, IMF, NATO, The Islamic State, The European Trading Scheme
- *Location* – East Asia, Europe, The Korean Peninsula, Silicon Valley, West Pacific, Baltics, The Baltic Sea, Harlem

All above examples are taken from the training data in Section 4.3.1.

5.3. Quality Analysis

The aim of the quality analysis is to assess the precision and recall values of each of the NER system candidates. Based on the results, we are able to select the most accurate NER system. This helps us to minimize the amount of incorrectly annotated named entities while executing the experiments from Section 4.2.

The analysis is executed on three independent test sets. Their statistics and properties are described in Subsection 5.3.1. Subsection 5.3.2 explains how the quality analysis is executed. Finally, the results together with factors influencing the evaluation are presented in Subsection 5.3.3.

5.3.1. Description of the test data sets

The evaluation is conducted on the following test sets:

- CoNLL-2003 Shared Task `eng.testb` test set (Sang and De Meulder, 2003) – This test set was created at the University of Antwerp. It was used during the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition challenge. The English data was taken from the Reuters Corpus[9].

[6]The entire list is presented in Appendix A.2.

[7]Corpus is available in the NLTK Downloader Explorer and here: `https://catalog.ldc.upenn.edu/LDC2005T09`.

[8]The full list is available at: `http://www.nltk.org/nltk_data/`.

[9]`https://trec.nist.gov/data/reuters/reuters.html`

- Wikipedia Gold Standard (Balasuriya et al., 2009) – This test set consists of 149 articles manually annotated articles from the May 22nd, 2008 dump of English Wikipedia. The articles describing named entities were selected at random, with a roughly equal proportion of article topics from each of the four CoNLL-03 classes (LOC, MISC, ORG, PER).

- Groningen Meaning Bank (GMB) Corpus[10] (Bos et al., 2017) – consists of public domain English texts with corresponding syntactic and semantic representations. The GMB corpus is developed at the University of Groningen. The corpus was initially annotated with external language technology tools (for segmentation, part-of-speech tagging, named entity tagging etc.), however, thanks to the collaborative editing by experts ("people with the required linguistic knowledge") incorrect annotations have been eliminated.

  3000 sentences have been randomly selected from this data set for the purpose of the quality analysis.

Properties of the test sets

This subsection provides statistics about the used test sets and discusses their properties influencing the evaluation and the results of the quality analysis.

NER system	Sentences	LOC	PER	ORG	Total
CoNLL-2003	3,684	1668 (34%)	1617 (33%)	1661 (33%)	4946
Wikipedia Gold Standard	1841	1013 (36%)	934 (33%)	895 (31%)	2842
Groningen Meaning Bank	3000	1312 (55%)	512 (21%)	583 (24%)	2407

Table 5.2.: Quality analysis: Properties of the test sets

Table 5.2 outlines the number of sentences as well as the number of named entities (together with their percentage shares) of each test set.

5.3.2. Quality Analysis Description

This section describes the execution process of the quality analysis. For each sentence i in the test set we do the following:

1. Based on the named entity annotations provided with each test set, we create a list of named entities together with named entity class labels for the sentence i. We call this list a reference list, as it contains the universal truth about NE labels.

2. We run the candidate NER system over sentence i and receive, thus, a list with hypothetical named entities together with their class labels recognized by the tested NER system.

[10] `https://gmb.let.rug.nl`

3. We calculate the precision and recall values for sentence *i* by comparing whether all named entities in the reference list are in the same form (string-based) in the hypothesis list. We check if named entity boundaries and class type match.

   Each hit is called a "true postitive" (TP). If a named entity is present in the reference list, but missing in the hypothesis list we call this a "false negative" (FN); if a named entity is present in the hypothesis list, but missing in the reference list we call this a "false positive" (FP). We calculate the precision and recall values with the formulas presented in Section 2.3.

4. Finally, we calculate the F-score with the formula presented in Section 2.3. In the F-score, we set beta to be one, as precision and recall are equally important and call it the F1-score in the later part of this work.

Label mapping Table 5.1 provides a list of named entity class labels recognized by each of the NER system candidates. As we decided to focus on Person, Location and Organization classes only, we filter out other labels. There are, however, labels, e.g. Geo-Political Entity, instances of which fall under one of the three chosen categories. In such case, they are mapped onto their hypernym (from the three chosen classes).

Such mapping is only made for spaCy's class GPE (Geo-Political Entity). This class is mapped onto Location.

5.3.3. Evaluation

This subsection presents the results of the quality analysis as well as outlines their interpretation.

Test set	NER system	Precision	Recall	F1-Score
CoNLL-2003	spaCy NER	73.57%	45.26%	56.04%
	NLTK NER	55.73%	46.05%	50.43%
	Stanford NER	92.42%	69.15%	**79.11**%
Wikipedia Gold Standard	spaCy NER	62.57%	46.64%	53.44%
	NLTK NER	50.00%	53.80%	51.83%
	Stanford NER	81.14%	57.84%	**67.54**%
Groningen Meaning Bank	spaCy NER	75.82%	54.23%	**63.23**%
	NLTK NER	64.64%	56.01%	60.02%
	Stanford NER	71.24%	50.32%	58.98%

Table 5.3.: Results of the quality analysis: Precision, Recall and F1-Score of tested NER systems

Factors influencing the interpretation of the results

The following facts about the tests sets may influence the interpretation of the results of the quality analysis:

1. Stanford NER has been largely trained on the CoNLL-2003 training corpus. This corpus was developed in 2003 and contains named entities which may no longer be up-to-date (new named entities emerge on a daily basis). Training data used for experiments in this work comes from the WMT news translation task from 2019 (compare with Section 4.3) and contains current proper names of people and organizations. As such, Stanford NER may have a decent quality on CoNLL-2003 test set, yet still under-perform while annotating WMT2019 news translation task data.

2. All three test sets are out-of-domain for spaCy and NLTK NER systems. However, Stanford NER has largely been trained on the CoNLL-2003 training corpus. Naturally, certain amount of named entities in the CoNLL-2003 `eng.testb` test set have already been seen by Stanford NER in the training data. This is a not zero-shot test set.

   Fu et al. (2014) confirm this observation about Stanford NER (English model). Stanford NER does not perform well on test sets, which have some difference from the training data. For example, its performance drops to 64.30% F1-Score on the Wall Street Journal test set.

3. The Groningen Meaning Bank corpus is not a manually created test set. However, it has been scrutinized, reviewed and corrected by the scientific community.

Interpretation of the results

Based on the results from Table 5.3, we draw the following conclusions:

1. Overall, the precision values are higher than recall values across all test sets and NER system candidates. This implies that the number of false positives is smaller than the number of false negatives while testing the NER system candidates. This is a positive phenomenon as we would prefer a NER system not recognize a named entity than to incorrectly annotate it. Incorrect and inconsistent annotation of the training data may mislead the network and make the learning process more challenging (compare with Section 4.1). Not annotating all named entities is not as harmful (yet still not desired).

2. Stanford NER has the highest F1-score for two test sets (CoNLL-2003 and Wikipedia corpus), whereas spaCy NER achieves the highest F1-score for the Groningen Meaning Bank corpus. The percentage differences in F1-scores between Stanford and other NER systems are higher for CoNLL-2003 and Wikipedia corpora than spaCy achieves for the Groningen Meaning Bank test set. This showcases Stanford NER's superiority over other NER system candidates.

   One has take into account, however, that CoNLL-2003 and Wikipedia tests sets are 17 or 10 years old (respectively) and Stanford may perform worse on up-to-date data from the WMT2019 news translation task.

3. SpaCy NER's F1-score is consistently higher than NLTK's over all three test sets.

5.4. Performance Analysis

The performance analysis focuses on measuring the execution time required to complete the quality analysis. Execution time plays a crucial role to annotate large amount of named entities in the training data within a reasonable time frame.

5.4.1. Experimental Setup

Due to the fact that the generation of source factors is implemented in Python, we use the Python-implemented NLTK class `nltk.tag.stanford`[11] to call Stanford NER in our evaluation landscape. This may have a negative influence on the execution time.

The performance analysis is executed on Ubuntu 16.04 with Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz processor with 4 cores. To measure the execution time of spaCy NER, NLTK NER, and Python-implemented Stanford NER extension Python package `time` was used.

5.4.2. Evaluation

Table 5.4 presents the results. Column "Sentences" describes the total number of sentences in each test set, column "Exec. time" describes the total execution time of the Python script, column "Time/1 sent." calculates the required time to annotate one single sentence ("sent." stands for "sentence"). The last feature will be used to make a prediction about the execution time required to annotate the training data set (approx. 2.1M sentences, according to Section 4.3).

Test set	NER system	Sentences	Exec. time in [s]	Time/1 sent.
CoNLL-2003	spaCy NER	3684	29.8	0.008
	NLTK NER	3684	37.5	0.010
	Stanford NER	3684	5988.3 ≈ 1h 40min	1.625
Wikipedia Gold Standard	spaCy NER	1841	14.1	0.008
	NLTK NER	1841	18.3	0.010
	Stanford NER	1841	2996.9 ≈ 50min	1.628
Groningen Meaning Bank	spaCy NER	3000	25.8	0.009
	NLTK NER	3000	32.5	0.011
	Stanford NER	3000	4907.3 ≈ 1h 22min	1.636

Table 5.4.: Results of the performance analysis: Execution times required to annotate the test sets

Interpretation of the results

Based on the results from Table 5.4, we draw the following conclusions:

[11] `http://www.nltk.org/api/nltk.tag.html#module-nltk.tag.stanford`

1. At first glance, Stanford NER's long execution times become visible. They are between approx. 160 to 200 times longer than for spaCy NER and NLTK NER. Such long annotation times make the use of the Stanford NER impossible. The requirement for the NER system candidates is to be able to annotate 2.1M sentences (from the training corpus) within a reasonable time frame. If Stanford NER's use necessitates 1.62s to annotate one sentence, it would take up to 990 hours (≈ 41 days) to annotate 2.1M sentences. Long execution times of the Python-implemented NLTK plug-in of Stanford NER are also confirmed in the its documentation[12].

   One possible solution to Stanford NER's long processing times would be to partition the training corpus into multiple subsets and annotate them in parallel. Another solution recommended by the official Stanford CoreNLP's documentation[13] is to set up a local Java server instead of starting a Java process each time the `StanfordNERTagger.nltk.tag` is called. According to this analysis[14], this should speed up the process 17 times (from 24 hours to 85 minutes), still not making Stanford NER acceptable for our use-case.

2. The estimated execution time, based on the results of the performance analysis, to annotate the training data set (Section 4.3) for spaCy is approx. 5 hours[15]. Moreover, spaCy NER and NLTK NER have a comparable execution speed.

5.5. Conclusion

The aim of this chapter was to select the most suitable NER system for annotating data used in the experiments described in Section 4.2.

Based on the results of the quality and performance analyses, we decide to continue with spaCy NER system. SpaCy is a fast and accurate tool and contains up-to-date language models (for English and German). In spite of its notable accuracy, Stanford NER is not able to annotate data in an efficient way. Moreover, spaCy NER system performs best on a most up-to-date test set (Groningen Meaning Bank), which is out-of-domain for all three NER system candidates.

[12] `https://stanfordnlp.github.io/CoreNLP/other-languages.html#python`

[13] `https://github.com/nltk/nltk/wiki/Stanford-CoreNLP-API-in-NLTK`

[14] `https://towardsdatascience.com/a-comparison-between-spacy-ner-stanford-ner-using-all-us-city-names-c4b6a547290`

[15] The actual execution time was approx. 6 hours. The difference may result from the fact that generating the source factors was not included in the performance analysis.

6. Evaluation

This chapter focuses on the evaluation of the experiments from Section 4.2. It starts by analyzing the properties of the test data set which is used to execute the evaluation. Later, it continues with the execution of three analyses: the automatic evaluation of the translation quality (measurement of the BLEU score), the automatic named entity evaluation and its human counterpart.

The first analysis aims at assessing the general translation quality, whereas the other two target to measure the translation quality of named entities. We present the algorithms: Automatic and Human Named Entity Evaluation used to conduct the before-mentioned analyses. We also discuss their properties and potential deficiencies.

Furthermore, we execute an assessment of the annotation quality of spaCy's NE German model and provide translation examples from the *annotated* model and the baseline. Finally, we execute the noise analysis to estimate the influence of the incorrect annotation by the NER system on the named entity translation quality.

6.1. Description of the test data set

This section presents the test data set from the WMT2019 news translation challenge *newstest2019*. Table 6.1 displays relevant statistics about the *newstest2019* test data set. SpaCy's model `en_core_web_sm`[1] is used to recognize and classify named entities.

Number of words	42,034
Number of NE	2,681
Ratio NE to all words	≈ 6.38 %
Number of sentences	1997
Number of sentences w/o NE	454
Number of sentences with other NE classes[2]	266
Number of sentences with NE	1277
Percentage of sentences with NE	≈ 63.95 %

Table 6.1.: Characteristics of the WMT2019 test data set

The percentage-based characteristics of the *newstest2019* test set are similar to those from training data (the latter ones are displayed in Table 4.5). *newstest2019* has identical content for both language pairs (English-German and English-Chinese) and contains 1997

[1] `https://spacy.io/models/en#en_core_web_sm`

[2] A full list of NE classes recognized by spaCy NER is outlined in Appendix A.2.

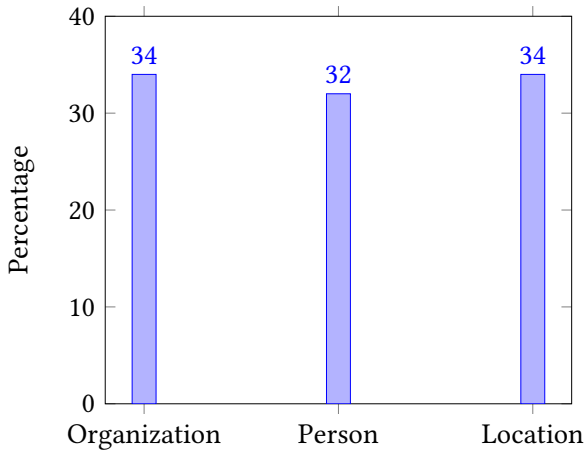

Figure 6.1.: Categorization of named entities in WMT2019 test data sets: En-De and En-Zh, in %

sentences, in which 63.95% of the sentences on the English side contain at least one named entity. The number of sentences with named entities in the test data set is approx. 10% higher than in the training data. Overall, 36% of sentences in this test data set do not contain any named entities. This is an important feature as we also wish to assess the performance of our trained models on a subset of *newstest2019* without any named entities (Section 6.2.2).

Figure 6.1 displays the percentage categorization of available named entities among three classes: Organization, Location and Person. There are 2681 named entity occurrences; 908 belong to the label *Location* (34% of all NEs), 870 to the label *Person* (32%) and 903 to the label *Organization* (34%). Each sentence with named entity occurrence contains, on average, approx. 2 NEs. There occurs almost identical division among named entity classes across three categories.

6.2. Evaluation of the general translation quality

This section reports the BLEU score (described in Section 2.9) for each of the models from Section 4.2.

Model no.	Label type	Variant	IOB	En→De	En→Zh
1	fine-grained	sum	no	**33.61**	26.29
2	fine-grained	concat 8	yes	33.11	**26.45**
3	fine-grained	sum	yes	33.07	26.26
4	coarse-grained	concat 8	yes	32.90	26.08
5	coarse-grained	sum	yes	32.70	26.34
6	Inline Ann. (fine-grained)		no	32.50	26.05
B	Baseline		no	32.60	26.29

Table 6.2.: BLEU scores on *newstest2019*

We perform the evaluation on the test data set *newstest2019* (described in Section 6.1). It is pre-processed in an identical manner as the training data (cf. Section 4.4). To assess the general translation quality, we calculate the BLEU score using the evaluation script *multi-bleu-detok.perl*[3] from Moses (Koehn et al., 2007). First, we transform the MT output into a tokenized text (we remove the BPE separator character `@@`). Further, we detokenize the MT output with *detokenizer.perl*[4] (Koehn et al., 2007) for En–De and use OpenNMT *detokenize* function to do the same for En–Zh.

Table 6.2 displays the results. Column "Model no." identifies each model with its id, "Label type" denotes whether specific ("fine-grained") or generic ("coarse-grained") NE labels are used; column "Variant" describes whether source factors are added ("sum") or concatenated ("concat") to the word embeddings; column "IOB" describes whether IOB tagging is used as a second source factor stream.

6.2.1. Evaluation of the BLEU scores on *newstest2019*

Based on the results in Table 6.2, we draw the following conclusions:

- Almost all models annotated with source factors show minor improvements w.r.t BLEU in comparison to the baseline. One Chinese model (model no. 4) has an insignificantly lower score than the rest.
- The fine-grained model with source factors added and no use of IOB tagging (model no. 1) seems to perform best for the translation from English to German and achieves around one BLEU point more than the baseline. While translating from English to Chinese model no. 2 achieves the highest BLEU score.
- Overall, German fine-grained models (models 1-3) achieve higher BLEU improvements than their Chinese counter-parts.
- There is no clear trend visible while assessing the "variant" column.

As the BLEU score only assesses the quality of NE translation indirectly, we do not deem it to be a reliable evaluation metric to assess the NE translation quality. As named entities affect only a small part of a sentence, we do not expect high BLEU variations. Hermjakob et al. (2008) make identical observations. In their work, they state that "general MT metrics such as BLEU, TER, METEOR are not suitable for evaluating named entity translation and transliteration, because they are not focused on named entities".

Consequently, the answers to the research questions presented in Section 4.1 cannot be resolved based on the BLEU scores. We answer them based on the results of an analysis which targets NE translation quality directly (Sections 6.3 and 6.4).

[3] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu-detok.perl`

[4] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/detokenizer.perl`

6.2.2. Evaluation of the BLEU scores on a test set without named entities

In this section we investigate how the NMT systems perform on test sets not containing any named entities. We are especially interested in determining how the *annotated* models perform. For this purpose, we calculate the BLEU scores for each of the models from Section 4.2 in the following experiment scenario.

The analysis is executed on the subset of the *newstest2019* test set which is created by filtering out all sentences with at least one named entity. SpaCy NER system (English model) is used to determine such sentences. We refer to this newly created test set as *nonNE-newstest2019* in later part of this work. *nonNE-newstest2019* test data set has 454 sentences (cf. with Table 6.1).

Model no.	Label type	Variant	IOB	En→De	En→Zh
1	fine-grained	sum	no	**32.11**	**19.97**
2	fine-grained	concat 8	yes	31.63	19.94
3	fine-grained	sum	yes	31.77	19.50
4	coarse-grained	concat 8	yes	31.14	19.57
5	coarse-grained	sum	yes	31.06	19.54
6	Inline Ann. (fine-grained)		no	30.88	19.05
B	Baseline		no	31.08	19.30

Table 6.3.: BLEU scores on *nonNE-newstest2019*

Based on the results from Table 6.3, we draw the following conclusions:

- Almost all models annotated with source factors show improvements w.r.t BLEU in comparison to the baseline. One German model (model no. 5) has an insignificantly lower score than the rest.
- The fine-grained model with source factors added and no use of IOB tagging (model no. 1) seems to perform best for the translation of both language pairs.
- Inline Annotation models for both language pairs perform slightly worse then the baseline.

In general, we state that external annotation does not influence the overall fluency and translation quality negatively in the *annotated* models.

6.3. Automatic Named Entity Evaluation

In this section we execute an automatic in-depth analysis of the NE translation quality with spaCy and Stanford NER (Finkel et al., 2005) systems. We use spaCy to evaluate German models and Stanford to do the same for Chinese models.

Test set: *random300* For this purpose, we randomly select 100 sentences from *newstest2019* containing at least one named entity for each of the three classes (*Person*, *Location*, *Organization*) on the English side of the corpus, in total 300 sentences. There is an equal percentage of named entities across all classes. We refer to this data set in later part of this work as *random300*.

6.3.1. Process description

The aim of the Algorithm Automatic Named Entity Evaluation (Algorithm 3) is to automatically assess the translation quality of named entities. The evaluation is called "automatic" due to the fact that we use a NER system to execute it. This evaluation takes place in the target language (German or Chinese). We use spaCy's NER German model[5] `de_core_news_md` for En–De and Stanford's NER (Finkel et al., 2005) Chinese model (4 classes) for En–Zh to locate named entities in the target sentences.

In the automatic NE evaluation we make use of the reference translations as they inter alia contain the correct translations of named entities. The following algorithm presents this evaluation process:

Algorithm 3 Automatic Named Entity Evaluation

For every reference translation i in the test data set, we do the following:

1. We annotate the reference sentence i with an external NER system (spaCy or Stanford NER) and create a list r with named entities in the reference translation. We do not enclose their class labels as they are not evaluated.

2. We check, with the Python function `find`, if each named entity in the list r is present at any position in the hypothesis (MT output). If yes, we define this case as a "hit", otherwise as a "miss". We store the number of "hits" and "misses" by adding them to the global variables: hit_g and $miss_g$.

Finally, we calculate the result according to the *NE match rate* formula:

$$NE\ match\ rate = \frac{hit_g}{hit_g + miss_g}$$

6.3.2. Results

This subsection presents the results of the automatic evaluation of the named entity translation quality of the MT output. It is executed on the *random300* test data set. The source factors annotating named entities are generated with the use of the algorithms presented in Section 4.5. All models from Section 4.2 are evaluated. Based on the results of this analysis, we are able to answer the research questions from Section 4.1.

[5] `https://spacy.io/models/de#de_core_news_md`

Tables 6.4 and 6.5 display the results. Column "Total" calculates the accumulated *NE match rate* for the three named entity classes.

Preliminary observations:

- At first glance, we see that the result values for En–De are significantly higher than for En–Zh (the total *NE match rates* for En–De are in the 70s, whereas the rates for En–Zh are in the mid-20s). We attribute this partly to the transliteration issues which emerge while translating from English to Chinese and, thus, occurring mismatch between the reference and hypothesis translation[6].
- In general, the baseline models show high performance (high *NE match rates*). This is attributed to the fact that a certain amount of named entities has already been seen by the network in the training data.

En→De							
Model no.	Label type	Variant	IOB	LOC	PER	ORG	Total
1	fine-grained	sum	no	73.68	70.11	61.79	69.89
2	fine-grained	concat 8	yes	72.87	**71.96**	63.41	70.67
3	fine-grained	sum	yes	**75.71**	70.85	**69.11**	**72.39**
4	coarse-grained	concat 8	yes	74.09	71.22	62.60	70.67
5	coarse-grained	sum	yes	75.30	71.22	65.04	71.61
6	Inline Ann. (fine-grained)		no	70.45	67.16	61.79	67.39
B	Baseline		no	74.09	71.59	60.16	70.36

Table 6.4.: Results of the automatic analysis on *random300* data set for En–De with spaCy NER, *NE match rate* in %

En→Zh							
Model no.	Label type	Variant	IOB	LOC	PER	ORG	Total
1	fine-grained	sum	no	**41.67**	20.07	31.62	24.41
2	fine-grained	concat 8	yes	33.33	**23.36**	36.76	**27.96**
3	fine-grained	sum	yes	**41.67**	20.44	33.09	25.12
4	coarse-grained	concat 8	yes	33.33	22.63	33.09	26.30
5	coarse-grained	sum	yes	33.33	21.90	**38.97**	27.73
6	Inline Ann. (fine-grained)		no	33.33	19.71	34.56	24.88
B	Baseline		no	33.33	18.98	35.29	24.64

Table 6.5.: Results of the automatic analysis on *random300* data set for En–Zh with spaCy NER, *NE match rate* in %

Based on the results from Tables 6.4 and 6.5, we answer research questions from Section 4.1. When we refer to the "model performance", we determine it by comparing the values from the column "Total" in the aforementioned tables.

[6] An explanation of the term "transliteration issues" is outlined in Section 6.3.3.

- **Research Question 1**

  Do annotated models achieve a better named entity translation rate in comparison to the baseline? Is named entity annotation in the form of source factors helpful?

  We observe improvements in named entity translation for En–De and En–Zh among almost all classes, showing that augmenting source sentences with NE information leads to their improved translation.

  Additionally, based on the fact that models 4-5 have higher total *NE match rates* than the baseline models, we state that the coarse-grained models perform better than the baseline. This finding indicates that the mere information that a word is a NE proves to be useful to the NMT system even if the class is not clearly specified.

- **Research Question 2**

  Is there a difference between fine-grained and coarse-grained annotation? Do specific named entity class labels contribute to an improved named entity translation?

  Augmenting the models with exact NE class labels (fine-grained case) seems to achieve, on some occasions, higher *NE match rates* in comparison to the coarse-grained case. This statement holds true for model pairs 3 and 5 (En-De) and 2 and 4 (En-Zh). In spite of this trend, there are a few exceptions, e.g. the performance of model 2 is weaker than model 5 (En-De). Identical observation can be made with model 3 being worse than models 4 and 5 (En-Zh).

  As model 1 is not trained with IOB tagging, we do not consider it while answering this question.

- **Research Question 3**

  Is IOB tagging helpful to translate compound named entities?

  Yes.

  There is no improvement in the models not using IOB tagging annotation (model 1). Their total *NE match rate* values are lower than that one of the baseline models. As such, IOB tagging, indicating compound named entities, proves to be an important piece of information for the NMT systems.

- **Research Question 4**

  How does inline annotation perform in comparison to named entity annotation with source factors?

  Inline Annotation does not deliver promising results, contrary to the findings of Li et al. (2018b), with the total *NE match rate* below that one of the baseline system (En–De) or insignificantly above (En–Zh).

- **Research Question 5**

  Does vector concatenation perform better than vector addition? Can a clear recommendation be defined for any given language pair?

Based on the performance of models 2-5, we do not recognize any clear trend. We conclude that model's variant (the fact whether it is trained while adding or concatenating word features to the word embeddings) is likely to be a hyper-parameter. Its value needs to be empirically determined to achieve the highest performance possible.

Validation of the *NE match rates* on Stanford NER

En→De							
Model no.	Label type	Variant	IOB	LOC	PER	ORG	Total
1	fine-grained	sum	no	76.25	76.14	60.00	73.70
2	fine-grained	concat 8	yes	75.62	77.16	64.62	74.88
3	fine-grained	sum	yes	**80.00**	**78.68**	**69.23**	**76.78**
4	coarse-grained	concat 8	yes	75.62	77.66	67.69	75.36
5	coarse-grained	sum	yes	77.50	76.65	**69.23**	76.48
6	Inline Ann. (fine-grained)		no	73.75	74.11	60.00	71.80
B	Baseline		no	78.75	76.65	60.00	74.64

Table 6.6.: Results of the automatic analysis on *random300* data set for En–De with Stanford NER, *NE match rate* in %

After having executed the automatic analysis with spaCy NER, we wish to validate the results of the En–De models with a second state-of-the-art NER system: Stanford NER. The analysis is conducted in an identical way as earlier and only the En–De models are analyzed. At the point of writing this thesis, spaCy does not provide a Chinese model. Table 6.6 presents the results.

Based on the results from Table 6.6, we draw the following conclusions:

- First, we observe that the overall *NE match rates* are higher than in Table 6.4. We attribute this phenomenon to the fact that Stanford NER recognizes a different set of NEs in the reference sentences than spaCy does. This, however, is not problematic as we are interested in the variations in *NE match rates* between the models. In general, there are no differences in the result trends and conclusions of the automatic analysis, regardless whether spaCy or Stanford is used to conduct it.

- All models trained with IOB tags translate NEs more accurately than the baseline model does. Again, fine-grained model trained with IOB tags and source factors added to the word embeddings achieves the highest *NE match rate*.

- The model trained without IOB tags has a lower *NE match rate* than the baseline re-confirming thus the usefulness of the IOB tags.

6.3.3. Drawbacks of the automatic named entity evaluation

This section outlines the drawbacks of the automatic named entity analysis.

1. **Transliteration issues**

   We observe that named entities in the reference translation (from the test set) are on some occasions translated according to the Chinese homophonic creation[7]. If the ML model decides to preserve a named entity from the source sentence in its verbatim form, the automatic evaluation is unable to recognize such cases and assesses them as "misses". These translations, however, may be correct as in certain cases it is acceptable to keep named entities in their original form. Transliteration may occur in the reference and hypothesis and causes, as a result, difficult to recognize mismatches. In the following, we present an example of transliteration causing evaluation difficulties:

Source	Mayorga claims Ronaldo fell to his knees after the alleged incident and told her he was " 99 percent " a " good guy " let down by the " one percent . "
Reference	马约尔加 称，罗纳尔多在指称事件发生后下跪对她说，自己“ 99% ”的时候都是“ 绅士”，而如今却被那“ 1% ”拖下水。
Hypothesis	Mayorga 号称Ronaldo 在据称事件发生后坠入其神职人员，并告诉她"99% "是" 良好服饰"，被"1% " 取代。

Table 6.7.: Example of a transliteration from English to Chinese; occurring in the reference only

Table 6.7 provides an example of a transliteration occurring in the reference translation only. Table 6.8 outlines a transliteration issue occurring in the hypothesis only.

Source	Speaking to the United Nations General Assembly , Foreign minister Walid al-Moualem said conditions in the country are improving .
Reference	外交部长Walid al-Moualem 在联合国大会上表示，叙利亚的条件正在好转。
Hypothesis	在联合国大会发言时，外交部长瓦利德·穆阿莱姆说，该国的条件正在改善。

Table 6.8.: Example of a transliteration from English to Chinese; occurring in the hypothesis only

Hypothesis translations from Tables 6.7 and 6.8 are evaluated as "misses", although their translation is linguistically correct.[8]

Transliteration issues partly explain the lower *NE match rates* across all models (1-6 and the baseline) for the language pair En–Zh in Table 6.5. The En–De automatic analysis may also be afflicted with errors of morphological origin. Their operating principles and effects on the automatic evaluation are identical with the transliteration issues.

[7] A list of Chinese homophones is available here: `http://hskhsk.pythonanywhere.com/homophones`.

[8] This observation is confirmed by a native speaker. Examples in Tables 6.7 and 6.8 origin from the human evaluation (Section 6.4).

2. **String-based search**

   We intentionally decide against annotating the hypothesis sentence with an external NER system to recognize named entities while executing the automatic evaluation. This would introduce even greater potential for incorrect named entity recognition. In detail, an external NER system may be inaccurate and may deliver on some occasions false positives, false negatives or provide incorrect named entity boundaries. We discuss the accuracy of a NER system in more detail in Section 6.4.4.

   Therefore, we decided to execute a string-based search in the hypothesis. While this evaluation form performs better, it also does not provide an optimal solution. With a string-based search we are incapable of determining if a named entity we are at a given point in time examining is in fact the one we wish to target.

   Let us illustrate this based on the following example, we observed while executing the automatic analysis:

Source	The George Washington Bridge is named after the first U.S. President: George Washington.
Reference	Die George Washington Brücke ist nach dem ersten U.S.-Präsidenten: George Washington benannt.
Recognized NEs (in ref.)	(George Washington Brücke, ORG), (U.S., LOC), (George Washington, PER)
Hypothesis	Die George Washington Bridge ist nach dem ersten U.S.-Präsidenten benannt: George Washigton.

Table 6.9.: Deficiencies of the string-based search

   Following the logic outlined in Algorithm 3, a string-based search searches for `George Washington Brücke`, and does not find this string in the reference. As a next step, it searches for `U.S.` and assess this check as a "hit". Later, it continues with `George Washington`. The person name `George Washington` has been misspelled by the NMT system. The automatic evaluation mistakenly finds this sub-string in `George Washington Brücke`. Final evaluation is: 1 hit, 2 misses. The correct evaluation, however, should be: 2 hits, 1 miss (It is acceptable to keep the name `Bridge` in a verbatim form).

3. **Inaccurate NER systems**

   In this paragraph, we describe how the inaccuracies of the NER system influence the automatic evaluation negatively.

   Each sentence in the test data set *random300* contains a named entity (according to the output of the spaCy's English model). There are, however, cases where, e.g., Stanford NER does not find any named entities in the Chinese reference translation producing, thus, false negatives. This occurs approx. 10 times out of 300 for Stanford NER and has not been observed while using spaCy NER system (German model). An example of a false negative error is presented in Table 6.10. Here, a named entity `Worcestershire`

is not recognized as "Location" in the reference translation. As a result, a correctly translated named entity in the hypothesis is not counted as a "hit".

Another example of a NER system inaccuracy is observed when false positives occur in the reference translations. This implies that the automatic evaluation is searching for words which are not named entities. An example of such occurrences is presented in Table 6.10. As `BST` is not a named entity, the automatic evaluation should not be assessing if such string is present in the hypothesis.

Source	Disorder broke out at HMP Long Lartin in Worcestershire at about 09:30 BST on Sunday and is ongoing.
Reference	Die Störung brach bei HMP Long Lartin in Worcestershire gegen 09:30 BST am Sonntag aus und dauert an.
Recognized NE (in ref.)	(HMP Long Lartin, ORG), (BST, ORG)

Table 6.10.: An example of how the inaccuracies of the NER system influences the automatic analysis

We address the drawbacks of the automatic analysis presented in this section by executing a human named entity evaluation in Section 6.4.

6.4. Human Named Entity Evaluation

In view of the drawbacks of the automatic evaluation (Section 6.3.3) and the fact that the NER systems are prone to delivering inaccurate results[9], we perform a human evaluation. Its aim is to provide a high level of certainty that annotating named entities with source factors leads to their improved translation. This section outlines the test data set, describes the evaluation rationale and presents the results.

6.4.1. Process Description

The human evaluation consists in recognizing named entities in the source sentence, comparing them to the corresponding named entities' translation in the MT output (in the hypothesis) and calculating the *NE match rate*. Note that the reference translation is used in a supporting role only and is not assessed directly. The human evaluation is conducted on the *random300* data set.

Algorithm 4 presents the logic according to which the human evaluation is executed:

[9] spaCy's German model has 83% F1-Score (`https://spaCy.io/models/de`) with a warning that it may "perform inconsistently on many genres", the same holds for Stanford NER: `https://nlp.stanford.edu/projects/project-ner.shtml`.

Algorithm 4 Human Named Entity Evaluation

For every source sentence i in the test data set, we do the following:

1. We search for named entities in the sentence i and create a list s with named entities in the source sentence. We do not enclose their class labels as they will not be needed.

2. We check if each named entity in the list s is *correctly translated* in the hypothesis sentence (from the MT output). If yes, we define this case as a "hit", otherwise as a "miss". We store the number of "hits" and "misses" by adding them to the global variables: hit_g and $miss_g$.

Finally, we calculate the result according to the *NE match rate* formula:

$$NE\ match\ rate = \frac{hit_g}{hit_g + miss_g}$$

We use the term "correctly translated" in Algorithm 4. As language may often be ambiguous, several correct translation variants may exist. When evaluating the translations from the MT output (hypotheses), we applied the following evaluation logic:

(i) if a named entity is in a different form in the hypothesis than the reference proposes or

(ii) a named entity is transliterated into or from Chinese,

but its form is still grammatically and semantically correct, its occurrence is counted as correct.

Test set We execute the human evaluation on the *random300* test data set. This is the same test set as used in the automatic analysis (Section 6.3). We choose this test set with an intent to be able to compare the results of the human and automatic analyses.

Compared models We compare the baseline and the best model (highest total *NE match rate* in Tables 6.4 and 6.5) for En–De and En–Zh and refer to them as *annotated* models. In detail, we evaluate the following models:

(i) for En–De: model no. 3 (compare with Table 6.4)

(ii) for En–Zh: model no. 2 (compare with Table 6.5)

Assessment guidelines The evaluation of En–De models is executed by a user with business-fluent near-native skills in German whereas the evaluation of En–Zh models is done by a Chinese native speaker. Human assessors discussed and unified the definition of each named entity class to ensure a homogeneous and fair assessment of each language pair.

Assessment style The human assessment is executed in a single-blind fashion. This implies that human assessors are unaware of the model's identity (model number).

6.4.2. The superiority of the human analysis

Section 6.3.3 outlines three deficiencies of the automatic analysis resulting from its nature of execution. This section outlines how the human analysis improves these issues. The following drawbacks are eliminated:

1. **Transliteration issues** Transliteration issues presented in point 1 result from the mismatch occurring between the reference and hypothesis translations. In the human analysis, however, the reference translation is no longer evaluated. Moreover, a native speaker is able to assess if a given named entity in the source sentence is translated into Chinese in accordance with the style common in the Chinese language. As such, transliteration issues in the human analysis are eliminated.

2. **String-based search** Errors resulting from the fact that the automatic analysis mistakes one named entity for another (as in Table 6.9) are eliminated. A human assessor recognizes which named entity is processed at a given point in time.

3. **Inaccurate NER systems** Evident miss-classification errors, such as false positives or false negatives (as presented in Table 6.10), are eliminated. Furthermore, thanks to the unification of definitions of each named entity class an objective evaluation is provided. Last but not least, as the evaluation is executed in a single-blind fashion, it is also deemed as impartial.

6.4.3. Results

Table 6.11 presents the results of the human evaluation. Column "Total" calculates the accumulated *NE match rate* for three named entity classes.

En→De							
Model no.	Label type	Variant	IOB	LOC	PER	ORG	Total
3	fine-grained	sum	yes	**93.02**	**83.52**	**78.01**	**85.17**
B	Baseline		no	89.77	82.05	70.92	82.14
En→Zh							
2	fine-grained	concat 8	yes	**73.85**	**67.04**	**64.27**	**68.05**
B	Baseline		no	71.43	61.90	57.35	63.24

Table 6.11.: Results of the human evaluation on *random300* data set, *NE match rate* in %

The *NE match rate* for human evaluation is higher than for its automatic counterpart. This is due to the fact that the deficiencies of the automatic evaluation (described in Section 6.4.2) are eliminated.

Based on the results from Table 6.11, we are able to reinforce our claims from Section 6.3.2.

- **Research Question 1**

  Do annotated models achieve a better named entity translation rate in comparison to the baseline? Is named entity annotation in the form of source factors helpful?

  We can state that the *annotated* models perform consistently better than the baseline and, in fact, the incorporation of external annotation in form of source factors into the source sentence leads to an improvement in named entity translation.

  There is an increase of 3.67% in the total *NE match rate* value for En–De and 7.61% for En–Zh. Further, we observe the greatest *NE match rate* improvement while translating organizations' names (+9.99% for En–De, and +12.07% for En–Zh).

6.4.4. The F1-score of the spaCy NER system on *random300* data set

While executing the human named entity evaluation, we also annotated false positives and false negatives of NE annotations in the reference, executing, thus, a quality check of spaCy NER on data from the news domain. The evaluation is executed on the *random300* data set and on the German model only. Precision and recall values are outlined in Table 6.12. The values are similar to the official ones provided in spaCy documentation[10].

Precision	84.43
Recall	85.93

Table 6.12.: Precision and recall values of the spaCy NER system, evaluated on the *random300* data set, in %

Observation Values in the above table lead to the conclusion that incorrect named entity annotation may occur relatively frequently in the training data. We hypothesize that annotating named entities with source factors may lead to better results if the training data is fully correctly annotated.

6.5. Translation examples

In this section we discuss our observations based on the human analysis and provide translation examples. We compare the translation outputs of the *annotated* model (model no. 3) and the baseline for the language pair English→German.

1. **Ignoring of low-frequency words**

   The use of source factors seems to alleviate the problem of ignoring low-frequency proper names as the *annotated* models appear to consistently react to named entity occurrence by producing a translation. The baseline, however, may ignore more complex named entities, producing, thus, an under-translation as in the *Alaska State Troopers* example in Table 6.13.

[10] `https://spacy.io/models/de`

Source	Palin, 29, of Wasilla, Alaska, was arrested (...) according to a report released Saturday by **Alaska State Troopers**.
Reference	Palin, 29, aus Wasilla, Alaska, wurde (...) verhaftet. Gegen ihn liegt bereits ein Bericht (...), so eine Meldung, die am Samstag von den **Alaska State Troopers** veröffentlicht wurde.
Annotated	Palin, 29 von Wasilla, Alaska, wurde (...) verhaftet (...), wie ein am Samstag von **Alaska State Troopers** veröffentlichter Bericht besagt.
Baseline	Laut einem Bericht von **Alaska**, der Samstag veröffentlicht wurde, wurde Palin, 29 von Wasilla, Alaska, (...) verhaftet (...).

Table 6.13.: Translation examples: The baseline model ignores the named entity (*Alaska State Troopers*) in the source sentence

2. **Under- and over-translation**

   Furthermore, source factors seem to guide the *annotated* models better (in comparison to the baseline) to prevent an over-translation, as shown in the *Home Depot* example or a miss-translation (*Gwyneth Paltrow's Goop*), both examples are in Table 6.14.

3. **Superfluous translation enforcement**

   On the other hand, a frequent cause of errors in the *annotated* models stems from the fact that organizations' or persons' names are translated verbatim instead of being preserved in their original forms, as in the *Francis/Franziskus* and *Giants/Giganten* example in Table 6.15. This problem concerns both the *annotated* model and the baseline. This behavior may not be desirable for persons' names, yet for organizations' names the desired output is dependent on the context and the translation language pairs.

Source	Saipov, 30, allegedly used a **Home Depot** rental truck (...).
Reference	Saipov, 30, hat (...) angeblich einen Leihwagen von **Home Depot** (...) benutzt (...).
Annotated	Saipov, 30, soll einen Mietwagen aus dem **Home Depot** benutzt haben (...).
Baseline	Saipov, 30, soll einen **Home Department Depot** Rental benutzt haben (...).
Source	The pair's business had been likened to **Gwyneth Paltrow's Goop** brand.
Reference	Das Geschäft der beiden war mit der Marke **Goop** von **Gwyneth Paltrow** verglichen worden.
Annotated	Das Geschäft des Paares wurde mit der Marke **Gop** von **Gwyneth Paltrow** verglichen.
Baseline	Das Geschäft des Paares wurde mit der Marke von **Gwyneth Palop** verglichen.

Table 6.14.: Translation examples: Under- and over-translations produced by the baseline

Source	The **Giants** got an early two-goal lead through strikes from Patrick Dwyer and **Francis** Beauvillier.
Reference	Die **Giants** hatten durch Treffer von Patrick Dwyer und **Francis** Beauvillier eine frühe Zwei-Tore-Führung.
Annotated	Die **Giganten** bekamen durch die Streiks von Patrick Dwyer und **Franziskus** Beauvillier ein frühes Ziel.
Baseline	Die **Giganten** erhielten durch die Streiks von Patrick Dwyer und **Francis** Beauvillier ein frühes Ziel.

Table 6.15.: Translation examples: superfluous translation enforcement

6.6. Estimation of the effect of the NER annotation quality

In Section 6.4.4 we state that spaCy NER achieves approx. 85% F1-score on the *random300* test data set. This value implies that a certain amount of named entities is incorrectly annotated in the training data and during inference time. In the same section we formulate a hypothesis stating that the NE translation improvement would be higher if the data were fully correctly annotated. In this section we aim at quantifying this claim.

The objective of this section is to perform an estimation of the effect of the NER annotation quality. As we are not able to improve the F1-score of the NER system[11], we knowingly and purposely introduce a predefined amount of annotation error into the training data, and wish to assess its effects on the NE translation rates. We call this analysis, in short, a *noise* analysis and execute it on En–De models only.

There are three types of errors committed by spaCy NER, we empirically observed during the human analysis:

1. False Positives – During the human evaluation, we noticed that this type of error seems to occur e.g. when a word is capitalized (e.g. “Sarahs Handy” is annotated as a person) or when all letters are capitalized (e.g. “10.00 AM CEST”, “CEST” is annotated as an organization). Most importantly, this type of error does not appear at random, e.g we do not observe its occurrence on regular verbs, pronouns, adjectives and adverbs. It would be incorrect to annotate a certain amount of words in a haphazard manner without investigating how probable it would be for spaCy to also annotate them in a false positive manner.

2. False Negatives – Based on our observations from the human evaluation, this type of error seems to occur when spaCy encounters a rare named entity, e.g. “East Baton Rouge” as an example of a location, and does not annotate it.

3. Incorrect Labels – This type of error occurs when a named entity is recognized, but it is incorrectly classified. This implies a false positive error of an entity and a false negative of the same entity but with a different class label. Intuitively, incorrect labels cause a double penalty as two types of errors are found.

[11] It would be possible to improve the F1-score of a given NER system, however such adaptation is out of scope of this work.

In general, it is challenging to develop a reliable heuristic which imitates the probability distribution of False Positive errors (error type no. 1) because of high linguistic complexity. We decide against annotating random words only. As such, we propose the following four experiments, which incorporate error type two and three only:

1. NA_{20} – 20% of NEs found by spaCy are not annotated imitating, thus, false negative errors (error type no. 2). *NA* stands for "not annotated".

2. IL_{20} – 20% of NEs have an incorrect label imitating, thus, error type no. 3. *IL* stands for "incorrect label". Instead of the classification from the NER system, the other two remaining labels are chosen with an equal probability. For example, if an entity is classified as an "ORG", it is annotated as either "PER" or "LOC" and so on.

3. IL_{40} – 40% of NEs have an incorrect label imitating, thus, error type 3. We are interested in discovering the effect of intensifying the error source for one experiment. For this purpose, we choose error type no. 3 and decide to double the amount of incorrectly annotated named entities.

4. $NA_{10} + IL_{10}$ – 10% of NEs are not annotated imitating, thus, false negative errors (error type no. 2). In addition, 10% of NEs have an incorrect label imitating, thus, error type no. 3. This experiment is designed to discover how the combination of two types of errors influences the NE translation rates.

The above experiments are conducted with an identical experimental setup (training corpus, pre-processing) as the experiments in Sections 6.2, 6.3 and 6.4. As we would like to investigate how robust the NE translation is with different quality of annotation, we introduce the noise to the training data set only. All models are evaluated on *newstest2019* with an identical NE annotation as in Sections 6.3 and 6.4.

Results and Evaluation

Table 6.16 presents the results of the *noise* analysis. The results of the En–De baseline model and the *annotated* model with the highest *NE match rate* are outlined towards the end of the table to enable an easier comparison. Their results are replicated from Tables 6.2 and 6.4.

En→De					
Experiment type	BLEU	LOC	PER	ORG	Total
NA_{20}	32.52	74.09	**73.06**	61.79	71.29
IL_{20}	32.47	74.90	71.96	61.79	71.14
IL_{40}	32.72	72.87	71.80	61.79	70.14
$NA_{10} + IL_{10}$	33.03	70.85	71.96	62.60	69.73
Baseline	32.60	74.09	71.59	60.16	70.36
Best model for En–De (model no. 3)	**33.07**	**75.71**	70.85	**69.11**	**72.39**

Table 6.16.: Results of estimation of the effect of the NER annotation quality on *random300* with spaCy NER for En–De, *NE match rate* in %

Observations:

- Both NA_{20} and IL_{20} have similar BLEU scores and experience a drop of approx. 1% in the total *NE match rates*. This would imply that both types of errors: no. 2 and no. 3 have a comparable effect on NE translation quality and lead to similar quality drops in translation.
- When comparing the total *NE match rates* of IL_{20} and IL_{40}, we state the following: Increasing the amount of incorrect labels by doubling their amount, results in an additional quality drop of 1 %. This may suggest a linear dependency.
- Based on the total *NE match rate* for $NA_{10} + IL_{10}$, we state that combining different types of errors, results in a significantly lower *NE match rates* than in the other experiments. This leads to the conclusion that the combination of different error types perplexes the network more than a higher amount of error of just one type. This is an interesting occurrence as a real NER system outputs errors of three different origins (error types no. 1, 2 and 3). This suggests that improving the F1-score of a NER system, may lead to a significant improvement in the *NE match rates* of the *annotated* models.

Overall, the *noise* analysis demonstrates that the improvement of the NER system annotation quality, would lead to an enhanced named entity translation while using the annotation method presented in this thesis. As a result, there is a potential for higher improvements in the *NE match rates* than those reported in the final human analysis in Section 6.4.

7. Conclusion and future work

This chapter summarizes the thesis with regard to achieved research results and outlines the general conclusions based on the findings of the analyses presented in this study. Furthermore, it considers shortly the strengths and limitations of our annotation approach. Finally, the chapter provides an overview of possible further research and improvements building on this work.

7.1. Conclusion

Challenges arising in the new digital era, require intensified international cooperation and collaborative work. A further intensification of intercultural exchange necessitates overcoming language barriers. In order to achieve this goal, automated translation solutions and large-scale Machine Translation systems are being made available to thousands of people over desktop and mobile solutions. The urge to deliver such platforms drives the research in the area of MT.

This thesis contributes to the area of named entity translation. Our work focused on establishing whether annotating named entities with the use of source factors leads to their more accurate translation. After executing a series of experiments, we can state the following:

1. The *annotated* models achieve higher BLEU scores than the baseline. As a result, we state that named entity annotation leads to higher BLEU values of the translated texts.

   Furthermore, we show that our annotation technique does not result in a worse translation performance than the baseline in the scenario where no named entities are present in the source sentence.

2. The automatic named entity evaluation demonstrates that the *annotated* models translate named entities with a higher translation quality than a regular NMT system trained with no external annotation. This conclusion is confirmed by validating the results of the automatic analysis on a second NER system.

3. Finally, based on the results of the human analysis, we can state that the *annotated* models perform consistently better than the baseline. There is an increase of 3.67% in the total *NE match rate* value for En–De and 7.61% for En–Zh. Further, we observe the greatest *NE match rate* improvement while translating organizations' names (+9.99% for En–De, and +12.07% for En–Zh).

All of the above observations lead to the conclusion that the incorporation of named entity annotation leads to their improved translation.

Limitations of the NE annotation approach

1. The application of the NE annotation approach during the inference time necessitates the development of pre-processing annotation pipelines the aim of which is to annotate named entities in the desired text with an external NER system prior to its translation. This may result in longer inference times depending on the implementation. According to the analysis conducted in Chapter 5, however, a state-of-the art NER system annotates a given text in an extremely performant way. Nevertheless, it shall be advised to measure the latency of the incorporation of named entity annotation (prior to its translation) into a NMT system in productive landscapes.

2. The quality of the NER system plays a crucial role during the annotation of named entities in the data as well as during the evaluation (cf. with Section 6.3). By establishing spaCy's F1-Score on *random300* test set during the human named entity analysis to amount to approx. 85%, we conclude that the accuracy of any NER system greatly influences the practicability of our approach. Therefore, the improvement of named entity translation is closely related to the improvement of NER systems.

   The *noise* analysis demonstrates the potential of the NE annotation method presented in this thesis. Should the annotation quality of the NER system greatly improve in the future, even higher *NE translation rates* shall be expected.

7.2. Future work

Further research may either focus on refining the system description and analyses presented in this study, or on addressing cross-domain topics in Machine Translation. In the following, we present potential further extensions of our work:

1. The statistical accuracy of the results of the automatic named entity analysis may be greatly improved if the inaccuracies of the NER system are fully eliminated. This may be achieved if a gold standard test set is used. In the context of this work, this implies annotating NEs in the *random300* test set manually instead of using spaCy or Stanford NER system to recognize NEs in text.

   Furthermore, the transliteration issues in Chinese occurring in the automatic named entity analysis can be eliminated if multiple correct Chinese translations were provided during the evaluation (e.g. including the Chinese homophonic translation or preserving a named entity in its verbatim form).

2. Another alternative approach worth investing is to use a hybrid NER system to annotate NEs in the data. The aim of pursuing such method is to improve the F1-score of NE annotation. As previously outlined, with raising annotation quality, the translation of named entities improves. Jiang et al. (2016) and Murugesan et al. (2017) outline ways of creating such a hybrid NER system. The output of multiple NER systems can be merged together by applying either a "union" operation or an "intersection" operation. The "intersection" improves the precision and reduces the recall, while applying the "union" operation has an adverse affect, it improves the recall and reduces the precision. In

general, the application of a hybrid NER system must be investigated in the current application scenario and potential benefits measured.

3. It would be interesting to execute identical experiments of NE annotation on a further number of language pairs. In doing so, the improvements in named entity translation rates can be averaged over a substantial number of experiments.

   Another possibility of reinforcing the findings of this thesis is to use different seeds to perform identical experiments several times and report the average scores. In doing so, we can further increase the confidence in the reported results.

4. Furthermore, a higher number of models could undergo a human evaluation. A clear benefit of this solution is providing an evaluation with high confidence scores. Unfortunately, this approach requires a high amount of expensive and time-consuming manual work. Therefore, efforts should focus on preferring automated solutions.

5. A natural extension of our approach is to use target factors annotating named entities in the target sentence. Examples of studies presenting target factors are Wilken and Matusov (2019) and García-Martínez et al. (2016). Such additional annotations would allow the network to create connections between the annotations on the source and the target side. This potentially may result in better *NE match rates*. It remains to be researched if more external annotation benefits the NE translation in the NMT systems.

6. Dinu et al. (2019) use source factors successfully to enforce custom terminology. Their method is more effective than a state-of-the-art implementation of constrained decoding. In light of these findings, future work may focus on the combination of NE annotation and terminology enforcement by providing multiple streams of source factors (= word features) into the input of the neural network.

   In general, further research may focus on the combination of the NE annotation method presented in this study with e.g. the annotation of morphological features, part-of-speech tags, syntactic dependency labels and others. Furthermore, it would be interesting to evaluate the influence of a higher number of external annotations with source factors on the MT quality and to assess where the limitations of such annotation with diverse word features are.

List of Abbreviations

ACL Association of Computer Linguistics.

ANN Artificial Neural Network.

BPE Byte Pair Encoding.

CNN Convolutional Neural Network.

CRF Conditional Random Fields.

DL Deep Learning.

GPU Graphical Processing Unit.

GRU Gated Recurrent Unit.

IE Information Extraction.

IOB inside-outside-beginning.

LSTM Long Short-Term Memory.

ML Machine Learning.

MT Machine Translation.

NE Named Entity.

NER Named Entity Recognition.

NLP Natural Language Processing.

NMT Neural Machine Translation.

POS Part-of-speech.

RNN Recurrent Neural Network.

SMT Statistical Machine Translation.

TDNN Time-Delay Neural Network.

Bibliography

Al-Onaizan, Yaser and Kevin Knight (2002).
"Translating named entities using monolingual and bilingual resources". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pp. 400–408.

Assal, Hisham, John Seng, Franz Kurfess, Emily Schwarz, and Kym Pohl (2011).
"Semantically-enhanced information extraction". In: *2011 Aerospace Conference*. IEEE, pp. 1–14.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016).
"Layer normalization". In: *arXiv preprint arXiv:1607.06450*.

Babych, Bogdan and Anthony Hartley (2003).
"Improving machine translation quality with automatic named entity recognition". In: *Proceedings of the 7th International EAMT workshop on MT and other language technology tools, Improving MT through other language technology tools, Resource and tools for building MT at EACL 2003*.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015).
"Neural machine translation by jointly learning to align and translate". In: *3rd International Conference on Learning Representations, ICLR 2015*.

Balasuriya, Dominic, Nicky Ringland, Joel Nothman, Tara Murphy, and James R Curran (2009).
"Named entity recognition in wikipedia". In: *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources (People's Web)*, pp. 10–18.

Banerjee, Satanjeev and Alon Lavie (2005).
"METEOR: An automatic metric for MT evaluation with improved correlation with human judgments". In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72.

Bawden, Rachel, Nikolay Bogoychev, Ulrich Germann, Roman Grundkiewicz, Faheem Kirefu, Antonio Valerio Miceli Barone, and Alexandra Birch (2019).
"The University of Edinburgh's Submissions to the WMT19 News Translation Task". In: *arXiv preprint arXiv:1907.05854*.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Jauvin (2003).
"A neural probabilistic language model". In: *Journal of machine learning research* 3.Feb, pp. 1137–1155.

Bos, Johan, Valerio Basile, Kilian Evang, Noortje Venhuizen, and Johannes Bjerva (2017).
"The Groningen Meaning Bank". In: *Handbook of Linguistic Annotation*. Ed. by Nancy Ide and James Pustejovsky. Vol. 2. Springer, pp. 463–496.

Brown, Peter F, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai (1992).
"Class-based n-gram models of natural language". In: *Computational linguistics* 18.4, pp. 467–479.

Callan, Jamie and Teruko Mitamura (2002).
"Knowledge-based extraction of named entities". In: *Proceedings of the eleventh international conference on Information and knowledge management*, pp. 532–537.

Chinchor, Nancy A. (1998).
"Overview of MUC-7". In: *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29 - May 1, 1998*. URL: https://www.aclweb.org/anthology/M98-1001.

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio (2014a).
"On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111.

Cho, Kyunghyun, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014b).
"Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734.

Collins, Michael and Yoram Singer (1999).
"Unsupervised models for named entity classification". In: *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

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

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa (2011).
"Natural language processing (almost) from scratch". In: *Journal of machine learning research* 12.Aug, pp. 2493–2537.

Crego, Josep, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, et al. (2016).
"Systran's pure neural machine translation systems". In: *arXiv preprint arXiv:1610.05540*.

Dale, Robert, Hermann Moisl, and Harold Somers (2000).
Handbook of natural language processing. CRC Press.

Dellaert, Frank, Thomas Polzin, and Alex Waibel (1996).
"Recognizing emotion in speech". In: *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96.* Vol. 3. IEEE, pp. 1970–1973.

Deselaers, Thomas, Saša Hasan, Oliver Bender, and Hermann Ney (2009).
"A deep learning approach to machine transliteration". In: *Proceedings of the Fourth Workshop on Statistical Machine Translation.* Association for Computational Linguistics, pp. 233–241.

Dinu, Georgiana, Prashant Mathur, Marcello Federico, and Yaser Al-Onaizan (2019).
"Training Neural Machine Translation to Apply Terminology Constraints". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3063–3068.

Doddington, George (2002).
"Automatic evaluation of machine translation quality using n-gram co-occurrence statistics". In: *Proceedings of the second international conference on Human Language Technology Research*, pp. 138–145.

Eberhard, David M., Gary F. Simons, and Charles D. Fennig (2020).
Ethnologue: Languages of the World. Twenty-third edition.

Emerson, Thomas (2005).
"The second international Chinese word segmentation bakeoff". In: *Proceedings of the fourth SIGHAN workshop on Chinese language Processing.*

Etzioni, Oren, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates (2005).
"Unsupervised named-entity extraction from the web: An experimental study". In: *Artificial intelligence* 165.1, pp. 91–134.

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning (2005).
"Incorporating non-local information into information extraction systems by gibbs sampling". In: *Proceedings of the 43rd annual meeting on association for computational linguistics.* Association for Computational Linguistics, pp. 363–370.

Forney, G David (1973).
"The viterbi algorithm". In: *Proceedings of the IEEE* 61.3, pp. 268–278.

Freitag, Dayne and Shahram Khadivi (2007).
"A sequence alignment model based on the averaged perceptron". In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 238–247.

Fu, Ruiji, Bing Qin, and Ting Liu (2014).
"Generating Chinese named entity data from parallel corpora". In: *Frontiers of Computer Science* 8.4, pp. 629–641.

Gage, Philip (1994).

"A new algorithm for data compression". In: *The C Users Journal* 12.2, pp. 23–38.

García-Martínez, Mercedes, Loïc Barrault, and Fethi Bougares (2016).
"Factored neural machine translation". In: *arXiv preprint arXiv:1609.04621*.

Gehring, Jonas, Michael Auli, David Grangier, and Yann Dauphin (2017a).
"A Convolutional Encoder Model for Neural Machine Translation". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 123–135.

Gehring, Jonas, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin (2017b).
"Convolutional sequence to sequence learning". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 1243–1252.

Ghader, Hamidreza and Christof Monz (2017).
"What does Attention in Neural Machine Translation Pay Attention to?" In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 30–39.

Goldberg, Yoav (2016).
"A primer on neural network models for natural language processing". In: *Journal of Artificial Intelligence Research* 57, pp. 345–420.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016).
Deep learning. MIT press.

Goyal, Archana, Vishal Gupta, and Manish Kumar (2018).
"Recent named entity recognition and classification techniques: a systematic review". In: *Computer Science Review* 29, pp. 21–43.

Grishman, Ralph and Beth Sundheim (1996).
"Message understanding conference-6: A brief history". In: *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Grundkiewicz, Roman and Kenneth Heafield (2018).
"Neural machine translation techniques for named entity transliteration". In: *Proceedings of the Seventh Named Entities Workshop*, pp. 89–94.

Gu, Jiatao, Kyunghyun Cho, and Victor OK Li (2017).
"Trainable Greedy Decoding for Neural Machine Translation". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1968–1978.

Ha, Thanh-Le, Jan Niehues, and Alexander Waibel (2017).
"Effective strategies in zero-shot neural machine translation". In: *arXiv preprint arXiv:1711.07893*.

Haizhou, Li, Zhang Min, and Su Jian (2004).
"A joint source-channel model for machine transliteration". In: *Proceedings of the 42nd Annual Meeting on association for Computational Linguistics*. Association for Computational Linguistics, p. 159.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016).
"Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hermjakob, Ulf, Kevin Knight, and Hal Daumé III (2008).
"Name translation in statistical machine translation-learning when to transliterate". In: *Proceedings of ACL-08: HLT*, pp. 389–397.

Hieber, Felix, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post (2017).
"Sockeye: A toolkit for neural machine translation". In: *arXiv preprint arXiv:1712.05690*.

Hochreiter, Sepp, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. (2001).
Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

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

– (1997b).
"LSTM can solve hard long time lag problems". In: *Advances in neural information processing systems*, pp. 473–479.

Huang, Fei (2005).
"Multilingual named entity extraction and translation from text and speech". PhD thesis. Citeseer.

Huang, Zhiheng, Wei Xu, and Kai Yu (2015).
"Bidirectional LSTM-CRF models for sequence tagging". In: *arXiv preprint arXiv:1508.01991*.

Jiang, Long, Ming Zhou, Lee-Feng Chien, and Cheng Niu (2007).
"Named entity translation with web mining and transliteration". In: *Proceedings of the 20th international joint conference on Artifical intelligence*, pp. 1629–1634.

Jiang, Ridong, Rafael E Banchs, and Haizhou Li (2016).
"Evaluating and combining name entity recognition systems". In: *Proceedings of the Sixth Named Entity Workshop*, pp. 21–27.

Kaiser, Lukasz, Aidan N Gomez, and Francois Chollet (2017).
"Depthwise separable convolutions for neural machine translation". In: *arXiv preprint arXiv:1706.03059*.

Kalchbrenner, Nal and Phil Blunsom (2013).
"Recurrent continuous translation models". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1700–1709.

Karimi, Sarvnaz, Falk Scholer, and Andrew Turpin (2011).
"Machine transliteration survey". In: *ACM Computing Surveys (CSUR)* 43.3, pp. 1–46.

Knight, Kevin and Jonathan Graehl (1998).
"Machine transliteration". In: *Computational linguistics* 24.4, pp. 599–612.

Koehn, Philipp (2005).
"Europarl: A parallel corpus for statistical machine translation". In: *MT summit*. Vol. 5. Citeseer, pp. 79–86.

– (2009).
Statistical machine translation. Cambridge University Press.

– (2017).
"Neural machine translation". In: *arXiv preprint arXiv:1709.07809*.

Koehn, Philipp and Hieu Hoang (2007).
"Factored translation models". In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pp. 868–876.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. (2007).
"Moses: Open source toolkit for statistical machine translation". In: *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pp. 177–180.

Koehn, Philipp and Rebecca Knowles (2017).
"Six Challenges for Neural Machine Translation". In: *Proceedings of the First Workshop on Neural Machine Translation*, pp. 28–39.

Kolen, John F and Stefan C Kremer (2001).
"Gradient flow in recurrent nets: The difficulty of learning longterm dependencies". In:

Kundu, Soumyadeep, Sayantan Paul, and Santanu Pal (2018).
"A deep learning based approach to transliteration". In: *Proceedings of the seventh named entities workshop*, pp. 79–83.

Kuru, Onur, Ozan Arkan Can, and Deniz Yuret (2016).
"Charner: Character-level named entity recognition". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 911–921.

Lakew, Surafel M, Mauro Cettolo, and Marcello Federico (2018).
"A comparison of transformer and recurrent neural networks on multilingual neural machine translation". In: *arXiv preprint arXiv:1806.06957*.

Lang, Kevin J., Alex H. Waibel, and Geoffrey E. Hinton (1990).
"A time-delay neural network architecture for isolated word recognition". In: *Neural networks* 3.1, pp. 23–43.

LeCun, Yann and Yoshua Bengio (1995).
"Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10, p. 1995.

Li, Jing, Aixin Sun, Jianglei Han, and Chenliang Li (2020).

"A survey on deep learning for named entity recognition". In: *IEEE Transactions on Knowledge and Data Engineering*.

Li, Xiaoqing, Jinghui Yan, Jiajun Zhang, and Chengqing Zong (2018a).
"Neural name translation improves neural machine translation". In: *China Workshop on Machine Translation*. Springer, pp. 93–100.

Li, Zhongwei, Xuancong Wang, Aiti Aw, Eng Siong Chng, and Haizhou Li (2018b).
"Named-Entity Tagging and Domain adaptation for Better Customized Translation". In: *Proceedings of the Seventh Named Entities Workshop*, pp. 41–46.

Loper, Edward and Steven Bird (2002).
"NLTK: The Natural Language Toolkit". In: *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pp. 63–70.

Luong, Minh-Thang, Hieu Pham, and Christopher D Manning (2015).
"Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421.

Luong, Minh-Thang, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba (2014).
"Addressing the rare word problem in neural machine translation". In: *arXiv preprint arXiv:1410.8206*.

Manning, Christopher D, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky (2014).
"The Stanford CoreNLP natural language processing toolkit". In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pp. 55–60.

Maučec, Mirjam Sepesy and Gregor Donaj (2019).
"Machine Translation and the Evaluation of Its Quality". In: *Natural Language Processing-New Approaches and Recent Applications*. IntechOpen.

Meur, Céline Le, Sylvain Gallinao, and Edouard Geoffrois (2004).
"Conventions d'annotations en Entités Nommées". In: *ESTER*.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013).
"Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781*.

Murugesan, Gurusamy, Sabenabanu Abdulkadhar, Balu Bhasuran, and Jeyakumar Natarajan (2017).
"BCC-NER: bidirectional, contextual clues named entity tagger for gene/protein mention recognition". In: *EURASIP Journal on Bioinformatics and Systems Biology* 2017.1, p. 7.

Muslea, Ion et al. (1999).

"Extraction patterns for information extraction tasks: A survey". In: *The AAAI-99 workshop on machine learning for information extraction*. Vol. 2. 2. Orlando Florida.

Nadeau, David and Satoshi Sekine (2007).
"A survey of named entity recognition and classification". In: *Lingvisticae Investigationes* 30.1, pp. 3–26.

Nadkarni, Prakash M, Lucila Ohno-Machado, and Wendy W Chapman (2011).
"Natural language processing: an introduction". In: *Journal of the American Medical Informatics Association* 18.5, pp. 544–551.

Newmark, Peter (1981).
Approaches to translation (Language Teaching methodology senes). Oxford: Pergamum Press., pp. 70–83.

Nouvel, Damien, Maud Ehrmann, and Sophie Rosset (2016).
Named Entities for Computational Linguistics. Wiley Online Library.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002).
"BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pp. 311–318.

Peddinti, Vijayaditya, Daniel Povey, and Sanjeev Khudanpur (2015).
"A time delay neural network architecture for efficient modeling of long temporal contexts". In: *Sixteenth Annual Conference of the International Speech Communication Association*.

Pereyra, Gabriel, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton (2017).
"Regularizing neural networks by penalizing confident output distributions". In: *arXiv preprint arXiv:1701.06548*.

Rama, Taraka and Karthik Gali (2009).
"Modeling machine transliteration as a phrase based statistical machine translation problem". In: *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pp. 124–127.

Ramshaw, Lance A and Mitchell P Marcus (1999).
"Text chunking using transformation-based learning". In: *Natural language processing using very large corpora*. Springer, pp. 157–176.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986).
"Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Sang, Erik Tjong Kim and Fien De Meulder (2003).
"Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147.

Sarawagi, Sunita et al. (2008).
"Information extraction". In: *Foundations and Trends® in Databases* 1.3, pp. 261–377.

Schuster, Mike and Kuldip K Paliwal (1997).
"Bidirectional recurrent neural networks". In: *IEEE transactions on Signal Processing* 45.11, pp. 2673–2681.

Sekine, Satoshi and Chikashi Nobata (2004).
"Definition, Dictionaries and Tagger for Extended Named Entity Hierarchy." In: *LREC*. Lisbon, Portugal, pp. 1977–1980.

Sennrich, Rico and Barry Haddow (2016).
"Linguistic Input Features Improve Neural Machine Translation". In: *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pp. 83–91.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016).
"Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725.

Shaalan, Khaled (2014).
"A survey of arabic named entity recognition and classification". In: *Computational Linguistics* 40.2, pp. 469–510.

Simon, Eszter (2017).
"The Definition of Named Entities". In:

Singh, Sonit (2018).
"Natural Language Processing for Information Extraction". In: *arXiv preprint arXiv:1807.02383*.

Sountsov, Pavel and Sunita Sarawagi (2016).
"Length bias in Encoder Decoder Models and a Case for Global Conditioning". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1516–1525.

Stahlberg, Felix (2019).
"Neural Machine Translation: A Review". In: *arXiv preprint arXiv:1912.02047*.

Stahlberg, Felix and Bill Byrne (2019).
"On NMT Search Errors and Model Errors: Cat Got Your Tongue?" In: *arXiv preprint arXiv:1908.10090*.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014).
"Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*, pp. 3104–3112.

Tang, Gongbo, Mathias Müller, Annette Rios, and Rico Sennrich (2018).
"Why self-attention? a targeted evaluation of neural machine translation architectures". In: *arXiv preprint arXiv:1808.08946*.

Tiedemann, Jörg (2012).
"Parallel Data, Tools and Interfaces in OPUS." In: *Lrec*. Vol. 2012, pp. 2214–2218.

Ugawa, Arata, Akihiro Tamura, Takashi Ninomiya, Hiroya Takamura, and Manabu Okumura (2018).
"Neural machine translation incorporating named entity". In: *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 3240–3250.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017).
"Attention is all you need". In: *Advances in neural information processing systems*, pp. 5998–6008.

Waibel, Alexander, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang (1989).
"Phoneme recognition using time-delay neural networks". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3, pp. 328–339. ISSN: 0096-3518. DOI: 10.1109/29.21701.

Wang, Yuguang, Shanbo Cheng, Liyang Jiang, Jiajun Yang, Wei Chen, Muze Li, Lin Shi, Yanfeng Wang, and Hongtao Yang (2017).
"Sogou neural machine translation systems for wmt17". In: *Proceedings of the Second Conference on Machine Translation*, pp. 410–415.

Wilken, Patrick and Evgeny Matusov (2019).
"Novel Applications of Factored Neural Machine Translation". In: *arXiv preprint arXiv:1910.03912*.

Wimalasuriya, Daya C and Dejing Dou (2010).
Ontology-based information extraction: An introduction and a survey of current approaches.

Wu, Dan, Daqing He, Heng Ji, and Ralph Grishman (2008).
"The effects of high quality translations of named entities in cross-language information exploration". In: *2008 International Conference on Natural Language Processing and Knowledge Engineering*. IEEE, pp. 1–8.

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. (2016).
"Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144*.

Yadav, Vikas and Steven Bethard (2018).
"A Survey on Recent Advances in Named Entity Recognition from Deep Learning models". In: *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2145–2158.

– (2019).
"A survey on recent advances in named entity recognition from deep learning models". In: *arXiv preprint arXiv:1910.11470*.

Yan, Jinghui, Jiajun Zhang, JinAn Xu, and Chengqing Zong (2018).
"The Impact of Named Entity Translation for Neural Machine Translation". In: *China Workshop on Machine Translation*. Springer, pp. 63–73.

Yonghui, W, M Schuster, Z Chen, QV Le, M Norouzi, W Macherey, M Krikun, Y Cao, Q Gao, K Macherey, et al. (2016).
"Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144*.

Zaanen, Menno van and Diego Mollá (2007).
"A Named entity recogniser for question answering". In: *Conference of the Pacific Association for Computational Linguistics (10th: 2007)*. Pacific Association for Computational Linguistics, pp. 317–324.

Zhou, Jie, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu (2016).
"Deep recurrent models with fast-forward connections for neural machine translation". In: *Transactions of the Association for Computational Linguistics* 4, pp. 371–383.

Ziemski, Michał, Marcin Junczys-Dowmunt, and Bruno Pouliquen (2016).
"The united nations parallel corpus v1. 0". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 3530–3534.

Zipf, George Kingsley (1946).
"The psychology of language". In: *Encyclopedia of psychology*. Philosophical Library, pp. 332–341.

A. Appendix

A.1. Validation data

Table A.1 presents the statistics about the WMT2019 news translation task development/validation data *newstest2018*. This data is used during the training of ML models (experiments) outlined in Section 4.2.

Metrics	En-De	En-Zh
Number of words in corpus	68,643	98,308
Number of NE	2,941	5,173
Relation: NE to all words	≈ 4.28%	≈ 5.26%
Number of sentences	2,998	3,981
Number of sentences without NE	928	970
Number of sentences with other NE classes	459	553
Number of sentences with NE	1,611	2,458
Percentage of sentences with NE	≈ 53.74%	≈ 61.74%

Table A.1.: Number of named entities in WMT2019 validation/development training datasets: En-De and En-Zh

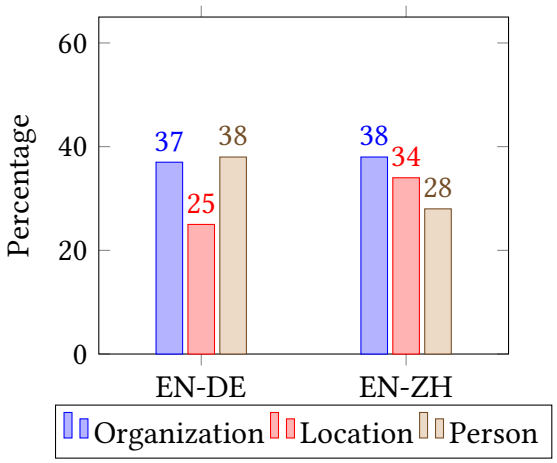

Figure A.1.: Categorization of named entities in development data sets: En-De and En-Zh, in %

A.2. spaCy's NER classes

Named Entity Class	Description
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK OF ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.
PERCENT	Percentage, including %.
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	"first", "second", etc.
CARDINAL	Numerals that do not fall under another type

Table A.2.: Recognized NE classes by spaCy NER