TÉCNICO LISBOA

Retrieval-based Adaptation For Machine Translation Applications Using Large Language Models And In-Context Learning

João Magalhães Rodrigues Sacadura Fonseca

Thesis to obtain the Master of Science degree in

Electrical and Computer Engineering

Supervisors: Dr. José Guilherme Camargo de Souza
Prof. Dr. André Filipe Torres Martins

Examination Committee

Chairperson: Prof. Célia Maria Santos Cardoso de Jesus
Supervisor: Dr. José Guilherme Camargo de Souza
Member of the Committee: Prof. Bruno Emanuel Da Graça Martins

November 2023

Declaration

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

This thesis focuses on one the most fascinating fields I have ever studied which is natural language processing, a subset of artificial intelligence. Firstly I want to thank my advisors, Professor André Martins and researcher José de Souza for awakening my interest in this area, believing in my capabilities and for their availability and helpful insights. I would also like to thank Unbabel for providing me the theme of this thesis, as well as my advisors and several bright and helpful people who I got to meet along the ride.

One can say this thesis is a one year work but for me it is much more. It is the culmination of a five year engineering degree that comes to an end. During this journey I had the support of many people which helped me become who I am today and for them I want to give a special thanks. These people include my family, who always supported me, my girlfriend Maria who showed me another side of the world, and my friends, some of which are detectives.

Abstract

Machine Translation (MT), the task of automatically translating a sentence from a source to a target language, has achieved remarkable progress recently, with the development of neural-based architectures. On the other hand, the work developed on Large Language Models (LLMs) in the past few years had a world-wide impact across many different industries, mainly due to their ability to achieve strong performance on a variety of tasks using the technique of *in-context learning*, without further training.

This thesis leverages the recent technique of *in-context learning* using LLMs on three MT applications, which are MT evaluation, terminology-constrained MT and automatic post-editing, by retrieving few-shot examples from a local fixed-sized datastore. The contents of the datastores as well as the prompts used for *in-context learning* are also analysed throughout this thesis.

We show that, despite not being explicitly trained with task-specific objectives, these models can have very competitive performance with the state-of-the-art architectures in some MT applications, with substantially smaller effort and cost compared to the industry standards. Moreover, we show these models demonstrate a strong capability to adapt to different scenarios with minimal effort, a skill unheard-of for existing MT systems.

Keywords

Machine Translation; Evaluation; Terminology; Automatic Post-Editing; Large Language Models; In-Context-Learning.

Resumo

Tradução automática, a tarefa de, automaticamente, traduzir uma frase de uma língua para outra, atingiu grande sucesso nos últimos anos, principalmente pelo desenvolvimento e avanço de arquiteturas neuronais. Por outro lado, os grandes modelos de linguagem tiveram também um grande desenvolimento que resultou num impacto de dimensão mundial em várias indústrias diferentes, principalmente pela sua capacidade de executa diferentes tarefas sem precisar de treino adicional, que tem o nome de aprendizagem contextual.

Esta tese tira proveito da recentemente descoberta técnica de aprendizagem contextual, utilizando os mesmos grandes modelos de linguagem para três tarefas dentro de tradução automática, que são a avaliação de traduções automáticas, a tradução de frases com restrições de terminologia e a pós-edição automática de traduções automáticas. Esta técnica usa exemplos que são retirados de um banco de dados guardado localmente com tamanho fixo. O conteúdo deste banco, assim como as mensagens enviadas para o grande modelo de linguagem para tirar proveito da aprendizagem contextual são alvo de estudo nesta tese.

Através desta tese mostramos que, embora estes modelos de linguagem não tenham sido treinados especificamente para as tarefas propostas, conseguem ter desempenhos fortes e a par das melhores arquiteturas em cada área, com muito menos custo e dificuldade de implementação, face ao que é costume nestas áreas. Além disto, mostramos que estes modelos demonstram uma forte capacidade de se adaptarem a diferentes cenários, como muito pouco esforço, uma habilidade previamente não existente na área de tradução automática.

Palavras Chave

Tradução Automática; Avaliação; Terminologia; Pós-Edição Automática; Grandes Modelos de Linguagem; Aprendizagem Contextual.

Contents

List of Figures

List of Tables

Acronyms

ANN	Artificial Neural Network
APE	Automatic Post-Editing
BART	Bidirectional Auto-Regressive Transformers
BERT	Bidirectional Encoder Representations from Transformers
BLEU	Bilingual Evaluation Understudy
biLM	Bidirectional Language Model
CBOW	Continuous Bag-of-Words
chrF	CHaRacter-level F-score
COMET	Crosslingual Optimized Metric for Evaluation of Translation
CoT	Chain of Thought
DA	Direct Assessment
ELMo	Embeddings from Language Model
FAISS	Facebook AI Similarity Search
GPT	Generative Pre-Trained Transformer
GRU	Gated Recurrent Unit
***k*NN-MT**	k-Nearest-Neighbour Machine Translation
LaBSE	Language-Agnostic BERT Sentence Embedding
LLaMa	Large Language Model Meta AI
LLM	Large Language Model
LM	Language Model
LSTM	Long-Short-Term-Memory
MQM	Multi-dimensional Quality Metrics
MT	Machine Translation

NLM	Neural Language Model
NLP	Natural Language Processing
NMT	Neural Machine Translation
PLM	Pre-Trained Language Model
QE	Quality Estimation
QEFV	Quality Estimation Feature Vectors
RNN	Recurrent Neural Network
RLHF	Reinforcement Learning from Human Feedback
SLM	Statistical Language Model
WMT	Workshop on Machine Translation

1

Introduction

Contents

1.1 Motivation

Translation is a vital part of the current society and an intrinsic part of every individual's lives. It allows humans to communicate in a global scale, to conduct business and grow our economy, to access and contribute to the constantly growing amount of information, to teach, to travel, among many things. Machine Translation (MT) allows us to scale, increase speed and performance (instant translations), as well as handle multilingual content, all in a cost effective manner when compared to the use of human translations.

Neural Machine Translation (NMT) has achieved remarkable progress in the last few years. Every year new approaches surpass the translation quality of previous methods. Recurrent Neural Networks (RNNs) (Elman, 1990) were introduced to handle sequential data. More recently, the transformer encoder-decoder architecture (Vaswani et al., 2017), built on the work of (Bahdanau et al., 2014) on sequence-to-sequence models, replaced previously used RNNs by the attention mechanism (Bahdanau et al., 2014), allowing these models to handle sequences of arbitrary size and capture context between generated words and a part of the source sentence. These advancements are sustained by the continuous increase in amount and accessibility of data as well as computational power.

One actively researched topic in the MT community is the ability of NMT models to maintain quality when translating content different than the content they were trained with, *i.e.*, in data with different contexts and topics than the ones used to train the model. This is a challenge because current approaches to MT do not deal well with out-of-domain generalisation (Saunders, 2021). Because of the complexity of NMT systems and the nature of their training process, these architectures tend to "memorise" training data, making it difficult to achieve same degree of performance in a real application, with unseen data from a different domain.

In order to mitigate this problem, there have been many approaches proposed with the technique of retrieval-augmented generation, *i.e.*, extending the current neural models with k-nearest neighbour approaches using fixed sized continuous-based indexed datastore (e.g. Facebook AI Similarity Search (FAISS) (Johnson et al., 2019)). This way the model has live access to millions of extra real quality examples and contexts (Khandelwal et al., 2020). By gathering large amounts of data of a certain domain, we can contribute to the solution this problem by allowing the NMT model to access the gathered contexts while generating each translation, at the word-level. This is known as k-Nearest-Neighbour Machine Translation (kNN-MT) (Khandelwal et al., 2021).

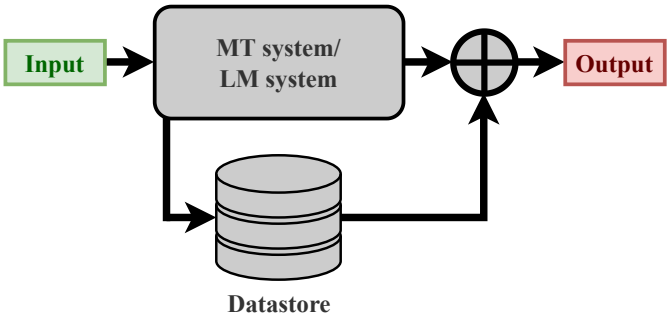


Figure 1.1: Diagram of a k-nearest-neighbour approach for language modelling or machine translation.

On a similar note, Language Models (LMs) have also achieved great progress for the tasks of language generation and understanding. Neural language models (Bengio et al., 2000,Mikolov et al., 2010,Kombrink et al., 2011) compute probability of word sequences through neural networks, such as the RNN. Recently, Pre-Trained Language Models (PLMs) were proposed, built on the transformer architecture (Vaswani et al., 2017) trained over a large-scale corpora. The use of datastores is not limited to MT. In fact, it was originally created for the language modelling task (Khandelwal et al., 2020). k-nearest-neighbour language models interpolate the probability distribution of PLMs with the contents of a datastore.

PLMs have been found to achieve better performance on downstream tasks the higher the parameter size and/or amount of training data (Kaplan et al., 2020), and not only that, when these models get very large, referred to as Large Language Models (LLMs), they gain surprising abilities, known as the emergent abilities (Wei et al., 2022). They have been shown to perform strongly perform on a series of complex tasks (Bubeck et al., 2023,Nori et al., 2023). As an example we have ChatGPT[1] that tailors LLMs, from the Generative Pre-Trained Transformer (GPT) collection (Radford et al., 2018), for dialogue, showing a revolutionary ability of conversation and in-context learning, causing world-wide impact.

Ever since the release of ChatGPT and its newest version powered by the new model GPT-4, the number of papers related to LLMs has greatly increased (Zhao et al., 2023), covering many different areas such as language modelling, translation, questions answering, text completion, text summarisation, sentiment analysis, classification, mathematical induction, code generation, among many others (Bubeck et al., 2023), through in-context learning.

Recently published research does not cover every task extensively and most of the evaluations are done with zero-shot prompting, *i.e.*, instructing the models to generate output for a specific task without providing any examples of the task. Motivated by this breakthrough and the current research this thesis focuses on two main questions:

- Can LLMs outperform, or compare, against the current state of the art in different applications of machine translation (MT evaluation, terminology constrained MT and Automatic Post-Editing

[1] https://openai.com/blog/chatgpt/

(APE))?

- How do LLMs prompted with k-shot in-context examples (providing k examples of the instructed task) perform against zero-shot scenario on different machine translation applications?

1.2 Contributions

This thesis proposes to further study the performance of the emergent closed LLMs, such as GPT-3.5 and GPT-4, in three widely researched machine translation applications: MT evaluation (with and without the use of a reference translation), terminology constrained MT and APE. In addition, we analyse the synergy of LLMs with fixed-sized datastores through many different methods of in-context learning using k-shot examples as depicted in Figure 1.2, as well as different methods for few-shot examples retrieval.

These models are used for their many applications through prompting techniques (Liu et al., 2021) which will also be experimented and analysed in this thesis for MT applications mentioned above. The choosing of these models takes several factors into consideration with the main ones being world-wide popularity, accessibility and pricing.

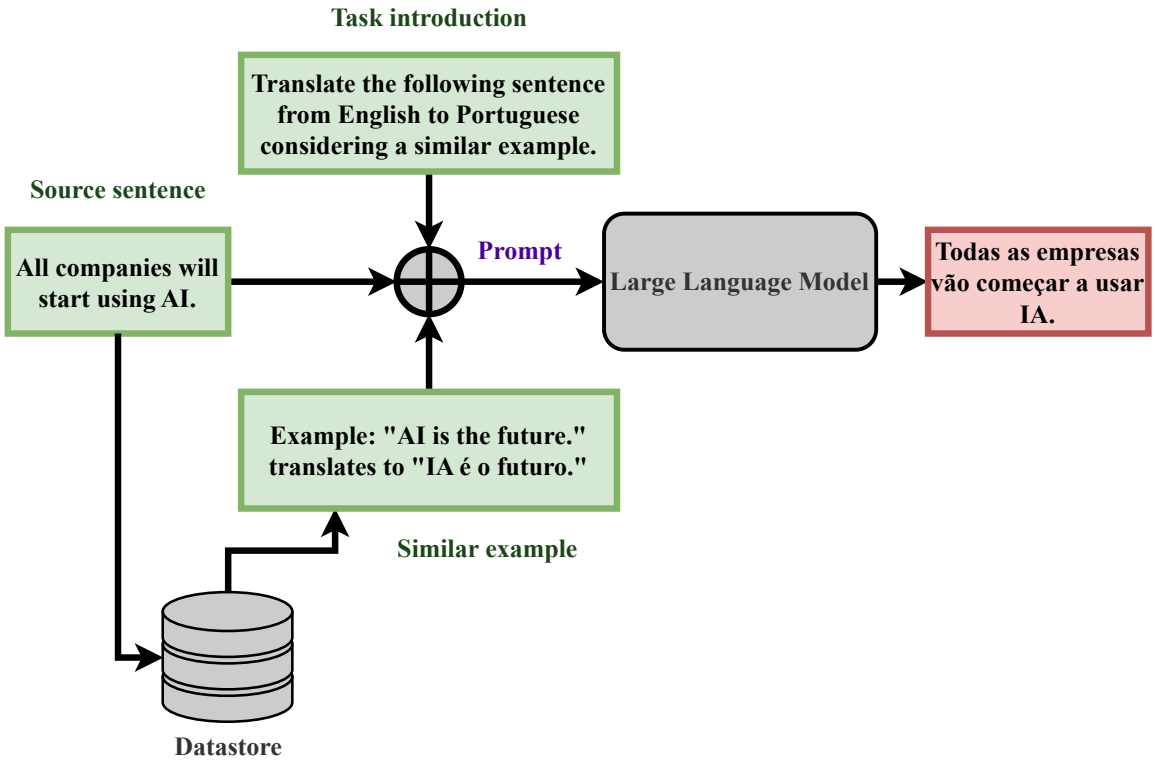


Figure 1.2: Diagram of proposed experiments involving few-shot augmented in-context learning prompts.

1.3 Thesis Outline

This thesis focuses on different areas of machine translation, as was already mentioned, with each area presented in a different chapter. Before, a background chapter is included to contextualise the areas in which this thesis focuses as well as the techniques and algorithms used. The structure is as follows:

- In Chapter 2, the machine translation and language modelling areas of research are presented as well as deep learning subjects that support them. This is important to understand the main results that support the Natural Language Processing (NLP), which is the main theme of this thesis, serving as an important context and deeper explanation of all the methods and algorithms used. This includes both history and state-of-the-art performance of machine translation, language modelling, machine translation evaluation, terminology constrained machine translation and automatic post-editing, which are the pillars of this thesis.
- In Chapters 3, 4 and 5, LLMs are thoroughly tested for MT evaluation, terminology constrained MT and automatic post-editing respectively with and without the use of fixed-sized datastores for in-context learning.
- Finally, in Chapter 6 the main conclusions of the previous chapters experiments are presented as well as routes for future directions of research are presented and justified with conclusions drawn from our own experiments.

2

Background

Contents

Deep learning is a subset of machine learning which develops techniques inspired by the working of the human brain. Currently, this is achieved by providing large amounts of data to neural network models that are able to detect patterns and, in this way, "learn" to perform specific tasks.

Deep learning architectures have achieved strong performance in many relevant tasks including data classification (for example, sentiment analysis of text or classification of an email as spam or not spam), regression (for example, prediction of house prices based on internal features), image generation, text/speech translation and generation, and many others, being widely researched and used in practical applications.

This thesis focuses on Machine Translation (MT), more specifically to its quality assessment, the appliance of terminology constrains, and automatic post-editing, all related to the field of Natural Language Processing (NLP).

2.1 Deep Learning Concepts

2.1.1 Artificial Neural Network (ANN)

ANNs are the foundation of complex deep learning architectures of current times. They are inspired by the human brain and are composed of an arbitrary number of layers with several artificial neurons all inter-connected. The artificial neuron is also inspired by the biological neuron and its earliest version dates back to 1943 via threshold logic (McCulloch and Pitts, 1943). While the threshold logic (called activation function) got replaced by more general functions, the artificial neuron concept remained the same.

Given a neuron p with t inputs, $x_1, \cdots, x_t$, a bias of x_0 and the correspondent weights, $w_{p0}, \cdots, w_{pt}$ (for the bias and the t inputs), its output is obtained as

$$y_k = g\left(\sum_{k=0}^{t+1} w_{pk} x_k\right), \tag{2.1}$$

where $g(.)$ is the activation function. There are many widely used choices for the activation function, such as the linear, sigmoid, hyperbolic tangent and rectified linear functions.

These neurons are then combined in a network, as shown in Figure 2.1 where the weights are optimised through a training process, to perform different tasks.

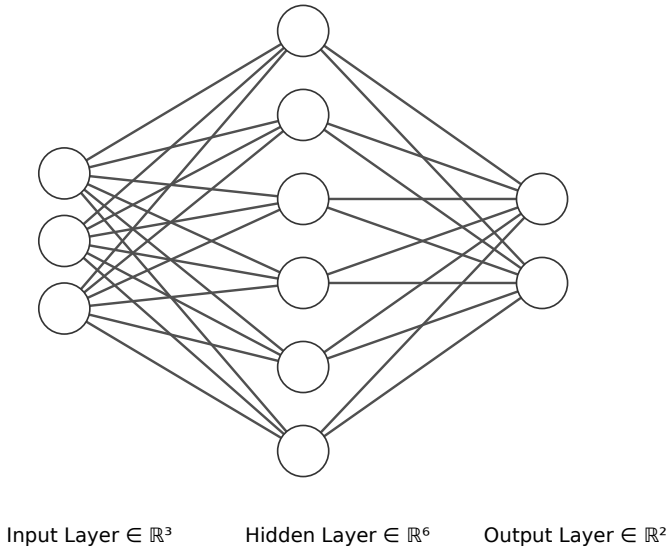


Figure 2.1: ANN architecture with three input layer neurons, a hidden layer with six neurons and two output neurons[1].

2.1.2 Recurrent Neural Network (RNN)

Several types of data are sequential by nature: DNA sequences, stock market returns, samples of sound signals and, of much importance to this thesis, words in sentences. RNNs (Elman, 1990) were introduced as an extension of ANNs to handle sequentiality. Because of their internal short memory, they can send information from one step to the next, as depicted in Figure 2.2. They are the foundation of the first machine translation systems.

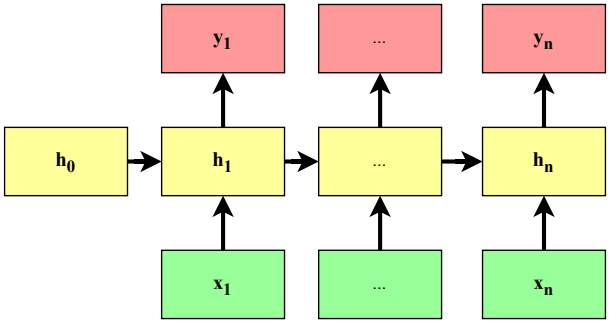


Figure 2.2: RNN architecture for n input tokens.

[1]Made with http://alexlenail.me/NN-SVG/index.html.

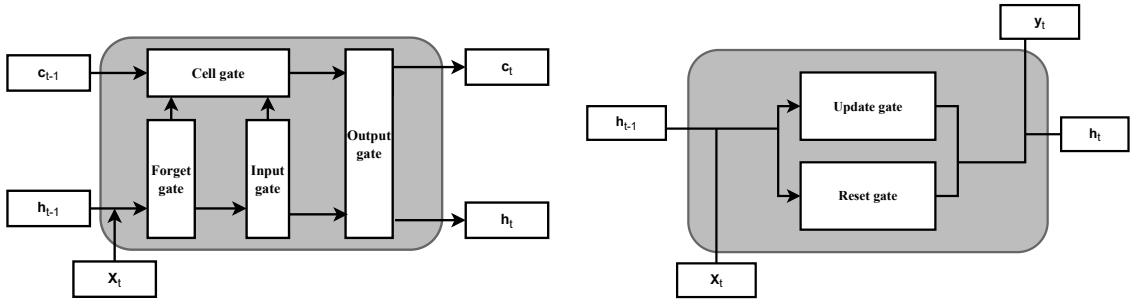
Each step is dependent of both the input $\boldsymbol{x}_t$ and the hidden state of the previous step, $\boldsymbol{h}_{t-1}$

$$\boldsymbol{h}_t = g(\boldsymbol{V}\boldsymbol{x}_t + \boldsymbol{U}\boldsymbol{h}_{t-1} + \boldsymbol{c}). \tag{2.2}$$

More specifically, calculation of $\boldsymbol{h}_t$ involves an affine transformation of both the input and the previous step state passing through a non-linear activation function g. The final activation step, and thus the final output, depend on all inputs, $\boldsymbol{x} = \boldsymbol{x}_1, ..., \boldsymbol{x}_n$, which leads to great expressive power in handling sequential data. In practice, we observe that RNNs tend to have a very short memory, forgetting older inputs and focusing on recent inputs. This is the reason why long dependencies are hard to capture, which is known as the vanishing gradient problem (Hochreiter, 1998).

To overcome this problem, Long-Short-Term-Memorys (LSTMs) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRUs) (Cho et al., 2014a) were created, both represented in Figure 2.3.

LSTMs introduce a memory cell called "cell state" (c_t) that maintains its value over time. Information from previous states can be added to these memories through gate mechanisms. Bidirectional LSTMs (Graves and Schmidhuber, 2005) combine two LSTMs, one to traverse the input sequence in its normal order (forward LSTM) and the other to do it in reverse (backward LSTM), thus capturing both left-to-right and right-to-left dependencies.



(a) LSTM architecture **(b)** GRU architecture

Figure 2.3: Representations of variations of the RNN structure that address the vanishing gradient problem.

GRUs are similar to LSTMs, but with the cell states removed. This way, the flow of information is now handled by the hidden states themselves, as represented in figure 2.3(b). These adaptive shortcuts are controlled by only two special gates, the update and reset gates, with the first one acting similarly to the input gate of an LSTM and the second controls which information to "forget".

Despite being introduced in 1997, LSTMs gained substantial relevance around 2014 and are still widely used today in a great number of applications: for example in 2016 with Google's *GNMT* architecture (Wu et al., 2016a), in 2017 with Amazon's *DeepAR* (Salinas et al., 2017) and in 2019 with Google's *Temporal Fusion transformer* (Lim et al., 2019).

2.2 Machine Translation (MT)

Machine Translation is the task of automatically translating text or speech from a source language into a target language.

Previous to 2014, statistical MT systems were used to find a certain target translation y for a given source sentence x. Formulating the problem using the noisy channel model (Koehn et al., 2003),

$$\hat{y} = \arg\max_{y} \ \mathbb{P}(x|y)\mathbb{P}(y). \tag{2.3}$$

This framework revolves around the idea that when a message is translated, it will have a distorted component, $\mathbb{P}(x|y)$. $\mathbb{P}(y)$ corresponds to the probability of a given sentence, which is computed by a language model which has to be designed and trained, while $\mathbb{P}(x|y)$ corresponds to the noisy channel.

Statistical MT systems were extremely complex, required substantial feature engineering, design was language dependent, consumed large amounts of data and required many different pipelines, such as language modelling (needed to compute $\mathbb{P}(y)$).

Neural Machine Translation (NMT) was proposed around 2013 (Kalchbrenner and Blunsom, 2013, Sutskever et al., 2014, Bahdanau et al., 2014) and the main difference with respect to previous approaches is its ability to train a machine translation model end-to-end, *i.e.*, without the need of a pipeline with different models. While there is space for experimenting with NMT pipelines, the underlying model is now more practical from a training and engineering perspective, requiring only one end-to-end model (a single encoder-decoder structure, also called sequence-to-sequence model) without the need of pipelining different models.

Sequence-to-sequence models

First sequence-to-sequence models (Cho et al., 2014b, Sutskever et al., 2014) use a Recurrent Neural Network (RNN) to encode the input sentence generating a state vector (Encoder) and another RNN that generates target sentence, word-by-word, taking the state vector as an input (Decoder).

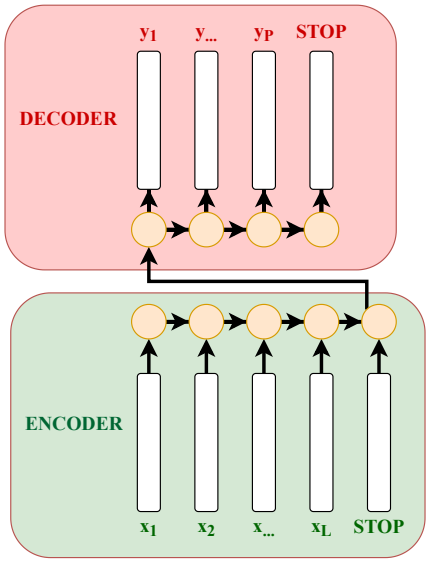


Figure 2.4: One of the first encoder-decoder architecture (Sutskever et al., 2014).

Some problems of this architecture is the fact that sentences of different lengths are mapped to the same vector length, which is a bottleneck. Furthermore, for long sentences, as mentioned earlier, the encoder may "forget" the initial words, substantially deteriorating translation of longer sequences.

NMT was revolutionary. Just two years after being introduced, it became the state-of-the-art approach to MT, mainly due to the appearance of the attention mechanism (Bahdanau et al., 2014) and more powerful architectures (such as the transformer (Vaswani et al., 2017)).

2.2.1 Transformer

The original transformer (Vaswani et al., 2017) is an encoder-decoder structure, with 6 stacked layers for both the encoder and decoder, illustrated in Figure 2.5. It is important to note that this structure is flexible and adaptable, as was already experimented in many applications. One example is a recent work (Kasai et al., 2020), in which transformers with a smaller number of decoder layers have been proposed, in order to increase parallelism and processing speed.

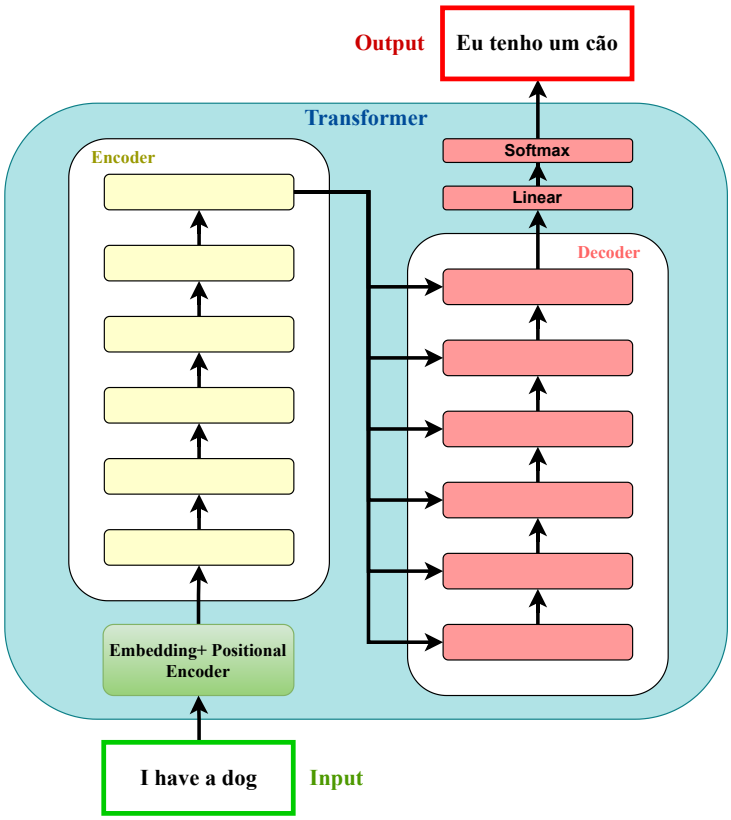


Figure 2.5: High level view of the transformer from (Vaswani et al., 2017).

The innovation behind the transformer lies on the removal of key components of previous state-of-the-art models, such as recurrence (through RNNs) and convolution structures, and replacing them entirely with attention mechanisms. The objective is to draw global dependencies between input and output. With this change, not only the quality of long and short sequences handling is increased but also allows for significantly more parallelization during training, achieving the new state-of-the-art in translation quality after being trained for a fraction of time of the previous best results.

The transformer was revolutionary for introducing the use of attention mechanisms (Bahdanau et al., 2014) in its components and has 3 main components, the **the embedding layer**, **encoder** and **decoder**, next described.

Embedding layer

The embedding layer is located between the first encoder and the input, and has the objective of transforming the text input to a structure accepted by the encoder, where words are represented in a continuous space. For this, a lookup table with learnt representations is used.

Computations in the encoder are parallel for all input tokens. Consequently, and since recurrence is not part of the topology of the transformer architecture, we also need a way to keep track of the position

of each token. To solve this, a vector called positional encoding is added to the input embeddings, which represents the position of each input text on the corpus. The positional encoder is defined as

$$\begin{aligned} PE_{(k,2i)} &= \sin(k/10000^{2i/d_{model}}) \\ PE_{(k,2i+1)} &= \cos(k/10000^{2i/d_{model}}), \end{aligned} \tag{2.4}$$

where d_{model} is the dimension of the output embedding space, k is the position of an object in the input sequence and i is the dimension index for the vector embeddings.

Encoder

The transformer architecture presents an encoder with 6 layers, all with an identical structure (although the weights are different).

Each encoder layer has a single self-attention layer followed by a feed forward neural network, as illustrated in Figure 2.6(a).

The self-attention block has the purpose of linking tokens from the same source language sentence. This way, for each input token, this layer calculates a relevance weight to each of the remaining tokens in the same sentence. For example, in the sentence "The dog ate a red apple", the word "dog" and "red" will both have a strong weight to the word "apple", since they are directly related in the meaning of the sentence.

We will now proceed to analyse the self-attention layers in more detail. The first step is to create 3 matrices, key ($\boldsymbol{K} \in \mathbb{R}^{L \times d_K}$), value ($\boldsymbol{V} \in \mathbb{R}^{L \times d_V}$) and query ($\boldsymbol{Q} \in \mathbb{R}^{L \times d_Q}$). These matrices are obtained multiplying the embedding vectors by projection matrices which are learnt in the training process. Given input embedding $\boldsymbol{x}_i \in \mathbb{R}^{d_{model}}$,

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^Q, \quad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^K, \quad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^V, \tag{2.5}$$

where $\boldsymbol{W}^Q \in \mathbb{R}^{d_{model} \times d_Q}$, $\boldsymbol{W}^K \in \mathbb{R}^{d_{model} \times d_K}$, $\boldsymbol{W}^V \in \mathbb{R}^{d_{model} \times d_V}$ and d_{model} is the embedding size which is assumed equal to the size of the output of each decoder.

These key, value, and query concepts are abstractions that represent different parts of the input sequence. Given these matrices, we start by computing an affinity score ($\boldsymbol{S} \in \mathbb{R}^{L \times L}$) between $\boldsymbol{Q}$ and $\boldsymbol{K}$. One widely used example is the dot-product affinity given by

$$\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{K}^T, \tag{2.6}$$

where it is assumed $d_Q = d_K$. The next step is to convert these scores into probabilities using the "softmax" function. However, the bigger the value of d_K, the larger $\boldsymbol{S}$ gets, the smaller the values of the "softmax" function gradients. To mitigate this, we normalise $\boldsymbol{S}$ by dividing it with $\sqrt{d_K}$.

The final self-attention score is obtained by multiplying these probabilities with $\boldsymbol{V}$

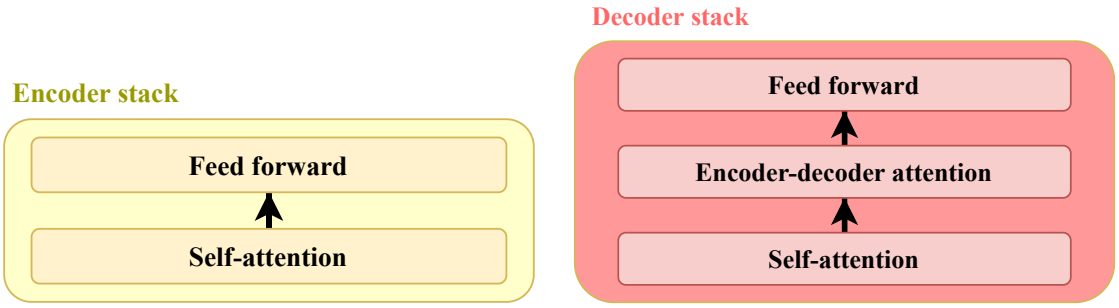
$$\boldsymbol{Z} = \textbf{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \textbf{softmax}\left(\frac{\boldsymbol{Q}.\boldsymbol{K}^T}{\sqrt{d_k}}\right).\boldsymbol{V}, \tag{2.7}$$

where the softmax transformation can be defined as

$$\text{softmax}(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}. \tag{2.8}$$

By repeating this self-attention process multiple times, *i.e*, by using multiple attention heads, it is possible and easier to capture multiple dependencies for each token. For example, by using multiple attention heads in the example above, the word "dog" could be linked to both the verb "ate" and the phrase "red apple" at the same time. In the original architecture (Vaswani et al., 2017), each self-attention layer of the original transformer has $h = 8$ heads. For each attention head j, we have 3 projection matrices $\boldsymbol{W}_j^K$, $\boldsymbol{W}_j^Q$ and $\boldsymbol{W}_j^V$ and a final attention score of $\boldsymbol{Z}_j$, with the same sizes as before, but with different values. All $\boldsymbol{Z}_j$ $(1 \leq j \leq 8)$ are concatenated in the end and multiplied by a final output projection matrix $\boldsymbol{W}^O \in \mathbb{R}^{h.d_V \times d_{model}}$

$$\begin{aligned} \boldsymbol{Z} = \textbf{Multi-head}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \textbf{Concat}(\textbf{head}_1, \textbf{...}, \textbf{head}_h)\boldsymbol{W}^O, \\ \text{where } \textbf{head}_j = \textbf{Attention}(\boldsymbol{Q}\boldsymbol{W}_j^Q, \boldsymbol{K}\boldsymbol{W}_j^K, \boldsymbol{V}\boldsymbol{W}_j^V). \end{aligned} \tag{2.9}$$



(a) Encoder stack structure

(b) Decoder stack structure

Figure 2.6: Constituents of encoder and decoder stacks.

Decoder

The number of decoder layers is also 6 in the original paper (Vaswani et al., 2017). Each decoder layer is composed of a self-attention, an encoder-decoder attention, and a feed-forward layer, illustrated in Figure 2.6(b).

The new element, the encoder-decoder attention, differs from the self-attention in that it receives the queries matrix $\boldsymbol{Q}$ from the previous decoder layer while the keys $\boldsymbol{K}$ and values $\boldsymbol{V}$ are obtained from the last layer of the encoder, illustrated in Figure 2.5. The idea is to have attention weights from each generated output word and their correspondent source inputs. For example if the sentence "A dog ate an apple" was translated to "Um cão comeu uma maçã" in portuguese, "dog" would have a high attention score towards "cão" since it's the correspondent direct translation.

The self-attention layer is also slightly different from the one in the encoder. Since machine translation is performed word by word, at a given time step we can only calculate attention towards the words that were already generated at a given time step. For instance, at the second time step we only have access to the current generated translation, for example: "Um cão _ _ _". To overcome this, future words are masked with the $-\infty$ value.

Layer normalization

One important detail to mention is that inside the encoder and the decoder, after every single sub-layer (self-attention, feed-forward and encoder-decoder attention) there is a normalization of its outputs in relation to that sublayer's inputs (note this is not represented in Figure 2.6).

To finalize the transformer architecture, after the decoder there is a linear transformation followed by a softmax activation function, from which we get the generated output at each time step.

2.2.2 Word Embeddings

Pre-trained Embeddings

The concept of representing words as vectors in a vector space exists since as early as the 1990s with models such as Latent Semantic Analysis (Landauer et al., 1998) and Latent Dirichlet Allocation (Blei et al., 2003). However, the concept of word embedding was first introduced in 2003 by (Bengio et al., 2003) who obtained them by training a neural language model and retrieving the models parameters. In 2008, their usefulness was established, being described as a highly useful tool when used in downstream tasks (Collobert and Weston, 2008), and in 2013 (Mikolov et al., 2013) proposed Word2Vec, a toolkit with learnt neural word representations at a lower computational cost, through two different architectures: Continuous Bag-of-Words (CBOW) and Skip-gram. GloVe (Pennington et al., 2014) was later introduced as an alternative to Word2Vec that trains on co-occurrence counts. The authors argued that Word2Vec and other local context window techniques underutilised the statistics in the training corpus.

These word representations have the great advantage of capturing both syntactic and semantic relations of large unlabelled corpora. However, they allow only for one representation per word, regardless

of the contexts in which it appears.

Contextual Embeddings

Contextual embeddings allow for multiple representation for a single word, depending on its context.

Embeddings from Language Model (ELMo) was introduced by (Peters et al., 2018) and it is a method to learn contextual word representations based on Bidirectional Language Models (biLMs). biLMs use context-free pre-trained embeddings, passing them through several bidirectional Long-Short-Term-Memory (LSTM) (Hochreiter and Schmidhuber, 1997, Graves and Schmidhuber, 2005) layers. However, according to (Radford and Narasimhan, 2018), these methods use a significant number of additional parameters, and their use of LSTM layers restricts the range of predictions they can make, especially for long-range dependencies. Consequently, the release of the transformer (Vaswani et al., 2017) model is the foundation of a new set of fine-tuning based models. These techniques build on a pre-trained language model, adding a few task-specific parameters, and then fine-tuning the entire model on the subsequent task. Latest architectures include OpenAI's Generative Pre-Trained Transformer (GPT) (Radford and Narasimhan, 2018) and RoBERTa (Liu et al., 2019), which is a more robust version of previously proposed Bidirectional Encoder Representations from Transformers (BERT) model (Devlin et al., 2018).

2.2.3 Machine Translation Evaluation

MT evaluation can be defined as the task of computing a quality score for translation hypothesis h of sentence s given a set of M reference translations $\mathcal{R} = \{v_1, ..., v_M\}$.

Quality Estimation (QE) is the subcategory of MT evaluation where the set of reference translations is empty, $\mathcal{R} = \varnothing$ (Blatz et al., 2004, Specia et al., 2018).

Reference-based Evaluation

The first method for automatic reference-based evaluation was to compare the number of n-grams[2] that appear in both the hypothesis and the reference, using metrics such as *precision* and *recall* to output a quality score.

These methods are very simple and fast to compute, not requiring any additional training, which makes them widely used in the area of MT evaluation. The most common n-grams based methods for are Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), CHaRacter-level F-score (chrF) (Popović, 2015). However, these approaches are unable to accurately cover long text dependencies, making them more focused on capturing lexicality and less precise capturing the semantics of a piece of text.

[2] n-grams are all the continuous sequences of n items (can be words or characters) within a piece of text.

Leveraging word embedding representations (Collobert and Weston, 2008, Mikolov et al., 2013) is another way to tackle the MT evaluation problem. With embeddings it is possible to compute a representation of an entire piece of text, whereas with n-grams this wasn't possible, making this method able to better capture similarities that go beyond the string matching level (Rei et al., 2020).

Modern neural-based methods, such as Crosslingual Optimized Metric for Evaluation of Translation (COMET), revolve around using cross-lingual encoder mechanisms to obtain these high-dimensional text representations which pass through a regressor block that computes the quality scores (Rei et al., 2020, Sellam et al., 2020, Pu et al., 2021). An illustration of such architecture is depicted in Figure 2.7.

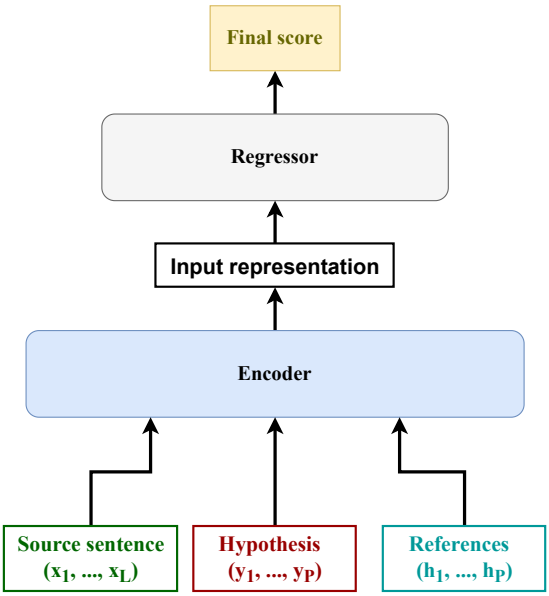


Figure 2.7: Modern architecture of latest neural based reference-based evaluation methods (Rei et al., 2020, Sellam et al., 2020, Pu et al., 2021).

Quality Estimation

QE systems design may differ according to their chosen input, which can be a group of phrases, a single phrase, or words. This task is of extreme importance in this field given the unavailability of reference translations in most real-life applications.

Word-level QE can be cast as a classification problem of attributing quality labels $\hat{y}_i \in \{\text{OK, BAD}\}$ for all machine translated words of a certain corpus. Sentence-level QE has the objective of attributing a single score $\hat{y} \in \mathbb{R}$ to each pair of source sentence and respective machine translation hypothesis.

Table 2.1: Word-level and sentence-level QE.

Target	Objective	Examples (EN→PT)	
Word	Classify as OK or BAD	dog → cão	⟹ OK
Sentence	Compute quality score $\hat{y} \in \mathbb{R}$	My dog loves to walk ↓ O meu cão gosta de andar	⟹ 0.89

The standard design for a QE system follows the predictor-estimator framework (Kim and Lee, 2016, Kim et al., 2017), in which a predictor block takes the hypothesis and the source language sentence and produces Quality Estimation Feature Vectors (QEFV). These are fed to the neural quality estimator block, outputting the final score. The framework is illustrated in Figure 2.8.

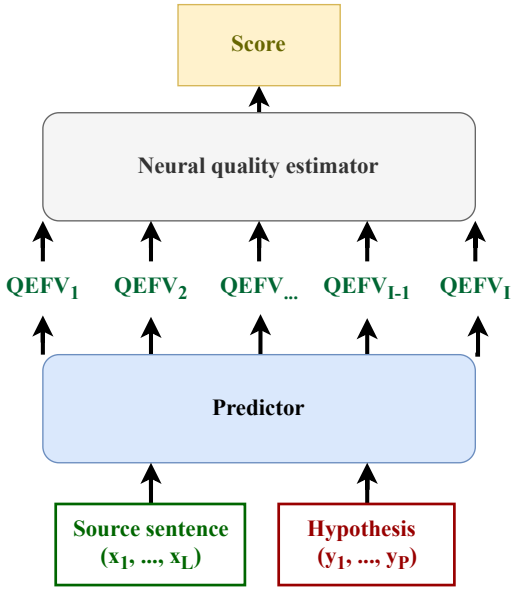


Figure 2.8: Predictor-Estimator framework (Kim and Lee, 2016, Kim et al., 2017).

Recent advanced QE models, including several previous studies (Wu et al., 2016b, Kepler et al., 2019, Fomicheva et al., 2020) take advantage of large pre-trained language models, usually encoder architectures such as XLM-R or RemBERT (Rei et al., 2022b) or even dual encoder (Heo et al., 2022), to process both the original text and its translation according to the evaluated model (predictor).

Evaluation of Metrics for Assessing the Quality of Machine Translation Systems

MT metrics achieved great success in the past few years, alongside the developments made in MT. There was an evolution to strictly lexical-based metrics, such as BLEU (Papineni et al., 2002) and chrF (Popović,

2015) which are based on n-grams analysis of segments to recent neural metrics such as COMET (Rei et al., 2020) and BLEURT (Sellam et al., 2020). These metrics are ranked according to their correlation to human judgements, mainly through Kendall Tau (τ), Pearson and Spearman correlations (Freitag et al., 2022) for segment-level evaluation, and also system-level pair-wise accuracy (Kocmi et al., 2021).

On the segment level, Kendall correlation is based on a pair-wise score, similarly to the system-wise accuracy. It has achieved many modifications over the years, which are different mainly due to the way ties between metric and humans are handled. Initially, Kendall's Tau-a (Kendall, 1938) was described as

$$\tau = \frac{\mathrm{C} - \mathrm{D}}{\mathrm{C} + \mathrm{D} + \mathrm{T}_h + \mathrm{T}_m + \mathrm{T}_{hm}}, \tag{2.10}$$

where "C" and "D" stand for Concordant and "Discordant" respectively, which are the sets of all the times a given metric agrees (or disagrees) with the order of systems established by human evaluators, T_h is the number of pairs tied only in the vector of human scores, T_h is the same for the vector of metric scores and T_{hm} is for ties happening in both human and metric scores. This correlation varies between -1 and 1 with a system that agrees 50% of the times with the human evaluation having a value of 0 for this correlation. This definition was later modified by (Stuart, 1953,Callison-Burch et al., 2011,Macháček and Bojar, 2013,Macháček and Bojar, 2014), mainly changing the ways ties are handled. One well-known drawback of Kendall's Tau is being susceptible to noise in gold pair-wise rankings

Pearson correlation tests linear fit with Multi-dimensional Quality Metrics (MQM) scores, which is a reasonable criterion since scores are expected to conform to a linear scale (for example, a translation with two minor errors is twice as bad as one with only a single error). Pearson has well-known drawbacks (Mathur et al., 2020) for example its sensitivity to outliers.

System-level accuracy (Kocmi et al., 2021) is described as the number of system pairs ranked correctly by the metric with respect to the human ranking divided by the total number of pair comparisons described by

$$\text{Accuracy} = \frac{|\text{sign}(\text{metric}\Delta) == \text{sign}(\text{human}\Delta)|}{|\text{all systems pairs}|}. \tag{2.11}$$

For each system pair, "metric(Δ)" is the difference between the both scores and "human(Δ)" is the difference between average human judgements. The accuracy is defined as the number of agreements between metric and human Δ divided by the total number of comparisons.

2.2.4 Terminology Constrained Machine Translation

Enforcing fixed translations of specific source terms is a recognised challenge within MT. In such scenarios, it becomes imperative to leverage external resources, known as glossaries, in order to meet certain requirements. Terminology constrained MT is the task of creating a translation T^{const} of a given source sentence S, where T^{const} is created using a set of constraints included in a glossary. The glossary can

include either specific untranslatable terminology, such as acronyms, or specific translations of a specific word.

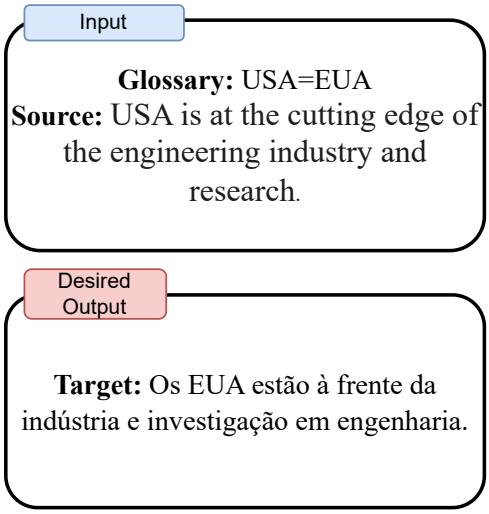


Figure 2.9: Example of terminology constrained MT.

Phrase-based statistical MT manipulates symbolic representations of source and target phrases, such as the XML markup implemented in (Koehn et al., 2007). Later on constrained decoding (Chatterjee et al., 2017) was implemented to address the same problem on NMT, by imposing such constrains directly at the decoding step. Constrained decoding also allows negative constraints, *i.e.*, a certain word must be avoided in the target sentence. However, these methods add substantial computational overhead to the inference step.

(Dinu et al., 2019) builds on this method by training a NMT transformer-based system to "learn" to apply terminology constraints when requested. This method improves the computational overhead at the cost of some quality in terminology inclusion percentage, and also requires the training of a transformer-based model.

2.2.5 Automatic Post-Editing (APE)

Despite the impressive evolution of machine translation, its quality is still far from perfect. In practical scenarios there is a post-editing step after the translation step, where modifications are made in order to improve translation quality. APE corresponds to the cases where the post-editing step is made without human intervention.

(Simard et al., 2007) introduced the first statistical automatic APE, which was a phrase-based machine translation on top of a rule-based MT system, in order to fix the repetitive errors which are natural of MT systems. In 2015 the first edition of the APE shared task was held at Workshop on Machine Translation (WMT)-15 (Bojar et al., 2015), presenting statistical-based approaches, with one

team including a RNN classifier to classify words in the automatic post-edit as good or bad. Later on, starting in WMT16, neural advances made it to the APE field, by training neural systems with `(src, mt pe)` triplets, corresponding to source, machine translation and post-editing. However, human-made post-editings were scarce which led to the creation and usage of artificial triplets (Junczys-Dowmunt and Grundkiewicz, 2016, Negri et al., 2018). After the transformer (Vaswani et al., 2017) was introduced, it was quickly shown to achieve state-of-the-art performance in APE. (Junczys-Dowmunt and Grundkiewicz, 2018, Tebbifakhr et al., 2018) use transformer-based approaches with two encoders for the source and machine translation, and one decoder for the post-editing. (Correia and Martins, 2019) fine-tuned a pre-trained BERT model on both encoder and decoder of an APE system, exploring transfer learning techniques. These transformer architecture based approaches have dominated the APE field ever since.

Recent progress in the area of language modelling, namely with the advancement of Large Language Models (LLMs) and their ability to perform on complex tasks without need for further training, also affected the APE field. (Raunak et al., 2023) tested the abilities of GPT-4 on APE in a zero-shot scenario.

2.2.6 *k*-Nearest-Neighbour Machine Translation (*k*NN-MT)

*k*NN-MT requires a standard NMT model, usually an attention-based encoder-decoder architecture (*e.g.* transformer (Vaswani et al., 2017)), already introduced in Section 2.2.1. The innovation of *k*NN-MT is that the output of the MT system is generated by augmenting the standard neural model with a non-parametric component, a datastore. The objective is to equip the process of assembling the translations with real-time access to sentences and contexts which are included as content of datastore. At each generation step, the system computes a distribution based on the k-nearest tokens inside the datastore and interpolates it with the neural model's distribution. A high-level architecture overview of a *k*NN-MT system is shown in Figure 2.10.

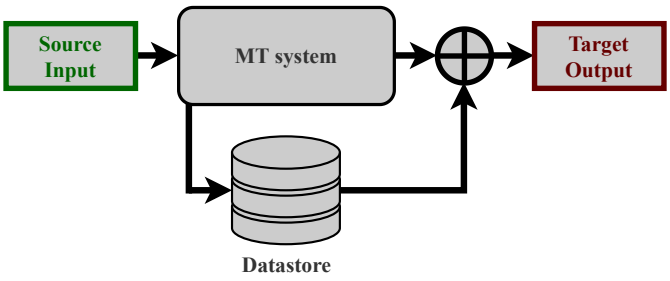


Figure 2.10: Diagram of a *k*NN-MT system.

Datastore Creation

The datastore is constructed offline, independently of the deployment and training of the neural model. The first step is to decide the contents of the datastore, which can be any type of parallel data. Key-value

pairs are created and inserted in the datastore, where the keys are high-dimensional decoder representations of the entire token sequence until token $(i-1)$, $\boldsymbol{f}(\boldsymbol{s}, \boldsymbol{y}_{1:i-1})$, and the values are simply the ground truth token $\boldsymbol{y}_i$. The datastore contents are defined as

$$(\mathcal{K}, \mathcal{V}) = \left\{ (\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{1:i-1}), \boldsymbol{y}_i), \forall \boldsymbol{y}_i \in \boldsymbol{y} \mid (\boldsymbol{x}, \boldsymbol{y}) \in (\mathcal{X}, \mathcal{Y}) \right\}, \tag{2.12}$$

where $(\mathcal{X}, \mathcal{Y})$ is a collection of parallel text collection (text in the source language, $\mathcal{X}$, and its translation in the target language, $\mathcal{Y}$).

Inference

At test time, we augment the decoder of the MT model by performing a linear interpolation, at each generated word, of the computed distributions of both the neural model, $p_{NMT}(\boldsymbol{y}_i|\boldsymbol{x}, \hat{\boldsymbol{y}}_{1:i-1})$ and the datastore $p_{kNN}(\boldsymbol{y}_i|\boldsymbol{x}, \hat{\boldsymbol{y}}_{1:i-1})$. Given an input token $\boldsymbol{x}$, the final probability for the ith generated word is

$$p(\boldsymbol{y}_i|\boldsymbol{x}, \hat{\boldsymbol{y}}_{1:i-1}) = \lambda \; p_{kNN}(\boldsymbol{y}_i|\boldsymbol{x}, \hat{\boldsymbol{y}}_{1:i-1}) + (1-\lambda) \; p_{NMT}(y_i|\boldsymbol{x}, \hat{\boldsymbol{y}}_{1:i-1}). \tag{2.13}$$

The parameter λ is of our choosing and dictates how much weight we attribute to the datastore word prediction. It is important to note that in order to compute the datastore's distribution, the neural model also needs to output $\boldsymbol{f}(\boldsymbol{x}|\hat{\boldsymbol{y}}_{1:i-1})$, the decoder output representation of that time step (word), in order to query the nearest neighbours.

To perform this datastore search, the Facebook AI Similarity Search (FAISS) library (Johnson et al., 2017) is used. It allows for efficient similarity search and also clustering of dense vectors, while allowing a reduction of 97% memory usage, using the product quantization (Jégou et al., 2011) method. The final kNN-MT distribution is obtained applying a softmax transformation of the negative distances (obtained using FAISS) over the vocabulary

$$p_{k\text{NN}}(\boldsymbol{y}_i|\boldsymbol{y}_{1:i-1}, \boldsymbol{x}) = \frac{\sum_{(\boldsymbol{k_j}, v_j) \in \mathcal{N}} \mathbb{1}_{\boldsymbol{y}_i = v_j} \exp\left(-d(\boldsymbol{k}_j, \boldsymbol{f}(\boldsymbol{y}_{1:i-1}, \boldsymbol{x}))/T\right)}{\sum_{(\boldsymbol{k_j}, v_j) \in \mathcal{N}} \exp\left(-d(\boldsymbol{k}_j, \boldsymbol{f}(\boldsymbol{y}_{1:i-1}, \boldsymbol{x}))/T\right)}, \tag{2.14}$$

where $\mathcal{N}$ is the set of k retrieved nearest neighbours, $\boldsymbol{k}_j$ denotes the key of the jth neighbour, v_j its value and T the softmax temperature.

It has been shown that the bigger the datastore size, the better contexts we're able to retrieve, and the better the system performs (Khandelwal et al., 2021). However this has a severe cost which is the degradation of the decoding speed. Ignoring the datastore creation process, which can be done offline, at each generation step the model has to search over the entire datastore (which can contain billions of contexts), causing a slowdown of up to eight times the decoding speed without a datastore. This is a severe bottleneck of this method, and thus, some methods have been recently proposed to improve

kNN-MT.

In order to improve decoding speed problems which were already introduced, several methods were proposed.

Chunk-based kNN-MT

(Martins et al., 2022) propose retrieving multiple tokens (chunks of tokens) at each retrieval step, reducing the number of retrieval steps, as well as using a neighbours cache to temporarily store the retrieved neighbours.

The decoder's output representation is similar to the one already mentioned in subsection 2.12, with the value now being c chunks of target tokens $\boldsymbol{y}_{t:t+c-1}$:

$$\mathcal{D} = \left\{([\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y_{1:i-1}}), ..., \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y_{1:i+c}})],\ \boldsymbol{y}_{i:i+c-1}),\ \ \forall i \in \boldsymbol{t}\ \ |(\boldsymbol{x}, \boldsymbol{y}) \in (\mathcal{X}, \mathcal{Y})\right\}. \tag{2.15}$$

The neighbours cache $\mathcal{M}$ is defined as

$$\mathcal{M} = \left\{(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}_{1:i+j}), y_{i+j})\ \forall 0 \leq j \leq c\ \forall \boldsymbol{y}_i\ |\ \boldsymbol{y}_{i:i+c-1} \in \mathcal{N}\right\}, \tag{2.16}$$

and it is a key-value memory. This way the model is able to access all tokens in the retrieved chunks, improving the ability to choose the tokens to retrieve.

Chunk-based kNN-MT leads to a decoding speed 4 times higher than the previously introduced method (Khandelwal et al., 2021), reducing the speed gap from alone neural machine translation model by a factor of 2.

Fast kNN-MT

Fast kNN-MT (Meng et al., 2021) revolves around reducing the search space by a large amount, searching only in a subset of the whole datastore.

First, similarly to previous kNN-MT methods, we obtain neural model's decoder representation of all source and target tokens, using the last layer of the encoder-decoder architecture. Next, for each source token we select the corresponding nearest neighbour tokens, with the search space being limited to the same token type as the initial source token. However, for some words, such as stop words, this search space can still be very large. To solve this, we have to also get the representation of each source token without any context. By doing this, we introduce a tunable parameter c which denotes the number of nearest neighbours for each token on the source side. We rank all candidates against this contextless representation of the source token and select the top c. The source side datastore, $\mathcal{D}_{source}$ is then defined as these c retrieved tokens for each source token.

On the target side, $\mathcal{D}_{target}$, is obtained by using FastAlign (Dyer et al., 2013) to map source tokens to correspondent target tokens, while abandoning mismatched tokens. At decoding time, we only search over a largely smaller datastore, $\mathcal{D}_{target}$, which is constructed offline, and perform standard kNN-MT (Khandelwal et al., 2021).

While addressing the decoding speed problem introduced by standard kNN-MT, a new problem of memory usage is introduced since we have to create a small datastore for each source target representation of a dataset with possibly billions of examples. To address this, product quantisation (Jégou et al., 2011) is used. This way, taking the WMT 2019 En-De dataset as an example, the memory size is reduced from 3.5TB to 108GB.

This method is two orders of magnitude faster than the standard kNN-MT and only 2 times slower than a single neural model without any non-parametric component.

2.2.7 Datastore Setup and Retrieval Methods

The FAISS (Johnson et al., 2017) library enables efficient similarity search. It allows us to build an index of a set of vectors and speed up the search times of that index, achieving amazing performance.

The simplest way to query a given vector is to measure the euclidean distance (or L2 distance) between the query vector and all other vectors present in the index. It is the most accurate method but very slow and does not allow the index to scale.

Partioning The Index

A popular approach to optimize our search from the basic setup is to partion the index into several index subsets called the Voronoi cells. The idea is to find the subset where our query vector is located, and limit the search to that subset, or extend it to nearby cells, thus reducing the scope of our search space substantially, producing only an approximate answer.

Product Quantisation

So far, all indexes have stored the vectors as wholes. This is acceptable for a small index, but quickly becomes a problem for larger datasets, which is commonly the case in neural architectures. To answer to this problem, FAISS comes with the ability to compress the vectors stored int he index, and to perform similarity search with the compressed indexes, using product quantisation (Jégou et al., 2011).

Product quantisation involves three steps: splitting a vector into several, equally-sized subvectors, assigning each subvector to its nearest centroid, and replacing the centroid values with uniqued IDs. The original high-dimensional vector is reduced to a much smaller vector of IDs that require up to 97% less memory usage.

The final version of the index used to store the data throughout this thesis is `IndexIVFPQ`, which performs the previously explained optimisations of dividing the index into multiple subsets and limiting the search space to some of those subsets, as well as product quantisation for the index vectors.

2.3 Language Modelling

As of today, machines are not able to grasp communication the same way a human does. Humans start to develop it in early ages and continuously advance their knowledge of it throughout their lives (Hauser et al., 2010).

Language modelling is the area of research in charge with creating and advancing models with the objective of creating communication ability, by capturing patterns, structures and relationships present in training text. Language models aim to model the generative likelihood of word sequences so as to estimate the probability of different symbols, tokens or token sequences. Representing a sequence of p words as $w_1, ..., w_p$, language modelling can be formulated as estimating the join probability of each word in the sequence

$$\mathbb{P}(w_1, ..., w_p) = \prod_{j=1}^{p} \mathbb{P}(w_j | w_{1:j-1}), \tag{2.17}$$

where the chain rule of probability was applied.

A fundamental difficulty associated with language modelling is data sparsity in addition with the *curse of dimensionality* (Bengio et al., 2003), which emphasises the likelihood of the model to be tested on different word sequences then the ones seen during training process. In fact, given a vocabulary size of $|V|$, the number of free parameters required to model a token sequence of length $|s|$ is $|V|^{|s|} - 1$.

Despite this, the literature on Language Models (LMs) is extensive and can be divided in four different areas, Statistical Language Models (SLMs), Neural Language Models (NLMs), Pre-Trained Language Models (PLMs) and LLMs according to (Zhao et al., 2023).

2.3.1 Statistical Language Models (SLMs)

SLMs also called n-gram Language Models (LMs) tackle the *curse of dimensionality* by taking advantage of word order and the fact that closer words in a sentence are statistically likely to have higher correlation, thus reducing the difficulty of the problem. An n-gram model approximates the full joint probability of the next word in a sequence, stated in equation 2.17, by the joint probability using only a context window of size n, *i.e.*, the last $n - 1$ words, formulated by

$$\mathbb{P}(w_j | w_{1:j-1}) \approx \mathbb{P}(w_j | w_{j-n+1:j-1}), \tag{2.18}$$

where the n-gram probabilities are obtained by maximum likelihood estimation, *i.e.*, obtaining the counts of the n-gram from a training corpus and normalising them. When facing with an unseen n-gram, smoothing techniques are used, such as Laplace smoothing, add-k smoothing, stupid backoff, Kneser-Ney smoothing (Chen and Goodman, 1999) as well as back-off models (Katz, 1987).

However, the n-gram assumption does not allow to capture longer dependencies than the context window, as well as the number of free parameters required is still very large, and grows exponentially with n. Moreover, these models do not account for the grammatical and semantic similarity between words.

2.3.2 Neural Language Models (NLMs)

NLMs model the joint probability in equation 2.17 through the use of neural networks such as Recurrent Neural Networks (RNNs). In a revolutionary way, (Bengio et al., 2003) introduced the concept of distributed word representations, already introduced in Subsection 2.2.2, through the use of an ANN that simultaneously learns the word feature vectors as well as the joint probability of word sequences as a function of these vectors.

This was the beginning of the use of language models for representation learning, going beyond the original objective of word sequence modelling. However, these initial approaches were based on lexicallity resulting in fixed word representations, meaning a single word would get the same representation no matter the context. This introduces the inability to distinguishing different semantic roles of the same word.

2.3.3 Pre-Trained Language Models (PLMs)

PLMs can create different representations for same word depending on its context (see Subsection 2.2.2).

The first advancements in this area come from Embeddings from Language Model (ELMo) (Peters et al., 2018). ELMo first pre-training a bidirectional Long-Short-Term-Memory (LSTM) network (Graves and Schmidhuber, 2005) and then fine-tuning it on specific downstream tasks. At each token j, the model retrieves $2L+1$ representations, $\left\{\boldsymbol{x}_j, \overrightarrow{\boldsymbol{h}_{ji}}, \overleftarrow{\boldsymbol{h}_{ji}} \quad | \quad i = 1, ..., L\right\} = \{\boldsymbol{h}_{ji} \quad | \quad i = 0, ..., L\}$, where $\boldsymbol{x}_j$ represents the initial fixed vector representation of word j and $\overrightarrow{\boldsymbol{h}_{ji}}, \overleftarrow{\boldsymbol{h}_{ji}}$ represent the outputs of the L layers of the forward and backward LSTM respectively (see Section 2.1.2). ELMo collapses all the $2L+1$ representations into a single vector. In a general way, for the specific task, the final representation is

$$\mathbf{ELMo}_j^{task} = \phi^{task} \sum_{i=0}^{L} q_i^{task} \boldsymbol{h}_{ji}, \tag{2.19}$$

where ϕ^{task} and q_i^{task} are task specific parameters learnt additionally to the ones of the bidirirectional LSTM.

Another step in this direction came from Bidirectional Encoder Representations from Transformers (BERT), which drew from the parallelizable transformer architecture using self-attention mechanisms (Vaswani et al., 2017). BERT's innovation lays in the pre-training of bidirectional language models using carefully designed tasks applied to large-scale unlabelled corpora.

This pioneering work paved the way for a multitude of subsequent studies that embraced the"pre-training" and "fine-tuning" approach. Following these advancements, a substantial amount of research has been dedicated to developing PLMs, which include Generative Pre-Trained Transformer (GPT) (Radford et al., 2018) and Bidirectional Auto-Regressive Transformers (BART) (Lewis et al., 2019), along with enhanced pre-training strategies (Liu et al., 2019).

GPT is a semi-supervised approach at language understanding tasks, combining unsupervised pre-training and supervised fine-tuning, with the transformer architecture (Vaswani et al., 2017) as the base model. For the pre-training a standard log-likelihood objective function[3] (a version of equation 2.17) with a *transformer decoder* (Liu et al., 2018) as the Language Model (LM) which is a variant of the original transformer architecture (Vaswani et al., 2017). After pre-training, the parameters are adjusted to a specific task by considering a labelled dataset where each instance consists of a sequence of input tokens $\boldsymbol{x}$ along with a label y.

BART (Lewis et al., 2019) is a denoising autoencoder and can be seen as generalisation of BERT and GPT. It uses the standard transformer architecture (Vaswani et al., 2017) with different activation functions and parameter initialisation. There are two main differences to the base model in the BERT architecture, with the first one being that each layer of the decoder performs additional cross-attention over the final layer of the decoder and there is no Artificial Neural Network (ANN) before final word prediction as there is in BERT. On the pre-training stage, BART is trained by corrupting documents and using a reconstructing loss function (cross-entropy between decoder's output and original document). The transformations used to pre-train BART are token masking, token deletion, text infilling, sentence permutation and document rotation. Unlike other models of its type, BART stands out because of its versatility. It is suitable to apply to any type of document corruption, and, in the extreme case where there is no source it also acts like a LM. Finally BART was fine-tuned on several different tasks: sequence classification, token classification, sequence generation and machine translation. Overall, BART has 10% more parameters than BERT.

As a further advancement to the BERT model, which proved to be an effective method for obtaining monolingual sentence embeddings for semantic similarity, (Feng et al., 2022) proposed Language-Agnostic BERT Sentence Embedding (LaBSE), which is an adaptation of BERT for obtaining cross-lingual sentence embeddings. LaBSE leverages a dual-encoder architecture which consists two transformer-based encoders

[3]Since probabilities are, by definition smaller than 1, there is a numerical underflow problem that happens when multiplying several of these probabilities together. Using log-probabilities improves on this by generating numbers not as small.

(Vaswani et al., 2017), one to encode the source sentence and the other one for the target sentence at training time. Before, both encoders are first pre-trained with masked language model (Devlin et al., 2018) and translation language model (Lample and Conneau, 2019) objectives respectively.

These pre-trained context-sensitive word representations proved incredibly effective and versatile as general-purpose semantic features. They significantly elevated the performance benchmarks for various tasks within the field of Natural Language Processing (NLP).

2.3.4 Large Language Models (LLMs)

Following the scaling law introduced by (Kaplan et al., 2020), researchers in NLP found that scaling PLMs regarding the number of parameters or data size often leads to improved model capacity on numerous tasks. Typically LLMs refer to transformer-based PLMs that contain dozens, hundreds or even thousands of billion of parameters such as GPT-3 (Brown et al., 2020), GPT-4 (OpenAI, 2023) or Large Language Model Meta AI (LLaMa) (Touvron et al., 2023a).

One of the main reasons that reinforces this scaling and that distinguishes LLMs from PLMs is the appearance of emergent abilities which can be defined as the abilities that are not present in small models but arise in larger models (Wei et al., 2022), or, in other words, performance rises substantially above random guessing in certain tasks after scaling the model over a certain point. The three main abilities that appear when scaling the model are *in-context learning*, *instruction tuning* and *step-by-step reasoning*.

In-context learning is introduced by the GPT-3 model and corresponds to the ability of LLMs to achieve surprising performance on downstream tasks by providing few input-label demonstrations related to the task, as shown in table 2.2. Currently, a well-built zero-shot prompt can often match, or come close, to the performance of few-shot learning. This fact in addition with the difficulty associated to engineering and retrieve good few-shot examples makes the zero-shot approach typically preferred.

Table 2.2: An example of *in-context learning* through two-shot learning .

Demonstrations:	
Revenues increased 20% over last quarter	**Positive**
Paying off my student loans will be extremely difficult	**Negative**
Test instance:	
I can tell the new product will have immediate positive impact	**?**

Instruction tuning is based on fine-tuning a LM to predict a certain response given a prompt which may, optionally, instruct the model about the task at hand, as was done with T0 (Sanh et al., 2022) and FLAN (Chung et al., 2022). Reinforcement Learning from Human Feedback (RLHF) is often used on top of instruction tuning using reward models to further align the models with human intent, without requiring a pre-defined response (Bai et al., 2022).

Standard ways of prompting suppose the output of the LLM to be in the final expected form as to

answer to the task that was instructed. As an alternative, *step-by-step reasoning* corresponds to a specific way of prompting a LLM to obtain not only the final expected output but also the correct reasoning to reach the final answer. LLMs have difficulties performing task that involve multiple reasoning steps, such as mathematical problems, and *step-by-step reasoning* as been shown to improve on this substantially. Different *step-by-step reasoning* methods include Chain of Thought (CoT) (Wei et al., 2023) (with an example shown in figure 2.11), self-consistency CoT (Wang et al., 2023) and *tree of thoughts* (Yao et al., 2023).

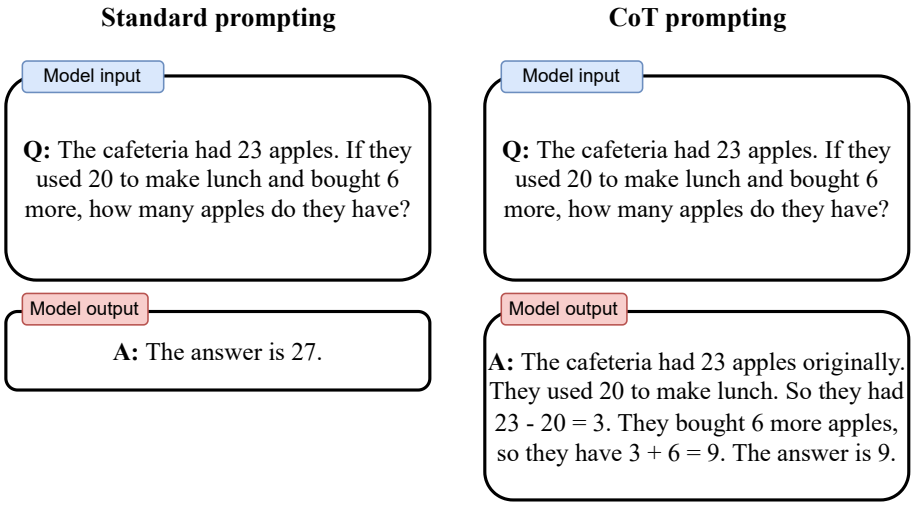


Figure 2.11: Example of CoT prompting technique that allows LLMs to tackle mathematical reasoning problem (taken from (Wei et al., 2023)).

Today's Large Language Models (LLMs)

The release of ChatGPT has sparked considerable excitement within the AI community due to its remarkable ability to engage in human-like conversations. Built upon the powerful GPT (Radford et al., 2018) model, ChatGPT has been fine-tuned to excel in conversational interactions. The success of GPT models hinges on two pivotal factors: training decoder-only transformer language models to predict forthcoming words and strategically increasing the size of these LMs. In figure 2.12 is an illustration of different GPT versions, some of which used throughout this thesis.

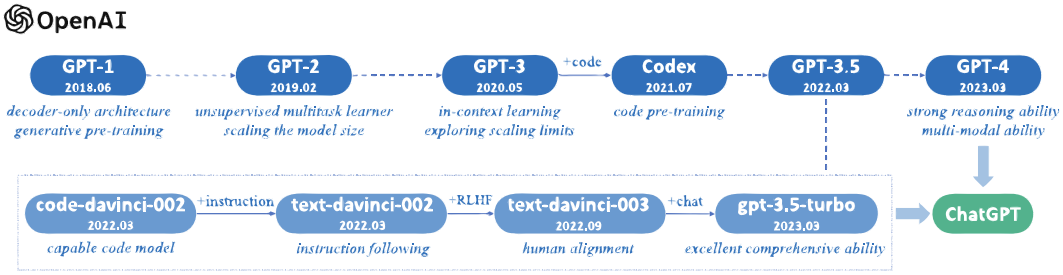


Figure 2.12: A brief illustration for the technical evolution of GPT-series models. The figure is mainly based on the papers, blog articles and official APIs from OpenAI. Here, solid lines denote that there exists an explicit evidence (*e.g.*, the official statement that a new model is developed based on a base model) on the evolution path between two models, while dashed lines denote a relatively weaker evolution relation. Image taken from (Zhao et al., 2023).

OpenAI's recent GPT models, such `gpt-3.5-turbo`, `text-davinci-003`, `GPT-4` shown in figure 2.12, as well as Google's recent models (such as BARD and PaLM (Chowdhery et al., 2022)) are only usable through closed APIs, meaning the that exact model architecture and parameter weights are not public, mainly due to safety concerns (Huang et al., 2023). Other organisations, such as Meta, believe that providing these models as open source, and consequently allowing the public the create their own modified versions, actually makes them "safer and better"[4]. Mark Zuckerberg, co-founder and chief executing officer of Meta also stated that "Open source drives innovation because it enables many more developers to build with new technology". A timeline with the most influential LLMs of the last few years is shown in figure 2.13.

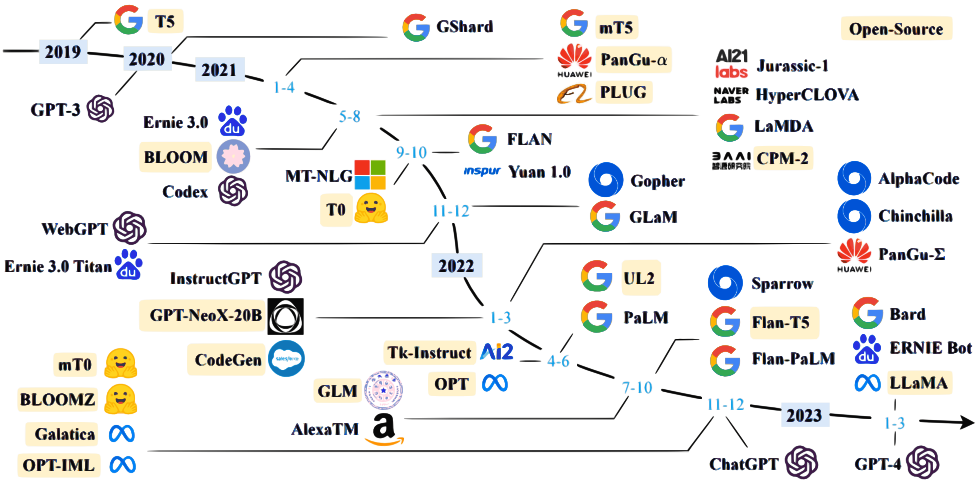


Figure 2.13: A timeline of existing LLMs (having a size larger than 10B) in recent years. The timeline was established mainly according to the release date (e.g., the submission date to arXiv) of the technical paper for a model. LLMs with publicly available model checkpoints are marked in yellow color. Due to the space limit of the figure, we only include the LLMs with publicly reported evaluation results. Image taken from (Zhao et al., 2023).

[4]According to Nick Clegg, Metas president of global affairs statement in the BBC Radio 4s Today programme.

3

Machine Translation Evaluation using Large Language Models

Contents

LLMs have been recently shown to be able to perform numerous varied tasks, such MT evaluation, with high quality, despite not being fine-tuned for this purpose. In this section we build on (Kocmi and Federmann, 2023) zero-shot MT evaluation experiments using LLMs, by introducing few-shot in-context examples. (Kocmi and Federmann, 2023) introduces `GEMBA`, a GPT-based metric for assessment of translation quality. They investigate nine versions of GPT models through zero-shot prompting, both with and without the use of references. The main conclusions are that some of the GPT-based metrics work well on document-level but lack segment-wise.

3.1 Experimental Setup

The used test set is the MQM 2022 human judgements for the English to German and Chinese into English language pairs[1]. It contains a total of 54 machine translation systems, most of them participants of the WMT 2022 general MT shared task (Kocmi et al., 2022). The segments of each language pair contain around 2k sentences from news, social, conversational, and e-commerce domains. The gold standard is human MQM ratings annotated by professionals who mark translation errors in each segment according to (Freitag et al., 2021).

We consider experiments with and without reference, based on the `GEMBA-DA` framework prompts used by (Kocmi and Federmann, 2023) which are scored from 0-100. All embeddings used as keys of the datastores for the in-context learning experiments were computed using the LaBSE model (Feng et al., 2022) and retrieved using the euclidean distance. To create the datastores and perform the nearest neighbour search, the library FAISS (Johnson et al., 2019) was used. The used LLM to output the quality scores was `gpt-3.5-turbo` with a cutoff date of June 2023[2].

All scores reported in the WMT 2022 Metrics shared task findings paper were reproduced using the official script (Freitag et al., 2022). The scores used were pair-wise accuracy (Kocmi et al., 2021) for system-level and Kendall's Tau-b (Freitag et al., 2022) for segment-level correlations.

3.2 Few-Shot Scenario

The base prompt used is shown in Table 3.1 and it is a variation of the one from (Kocmi and Federmann, 2023), where elements within curly brackets are substituted by the specific experiment specific information (as an example "{source_lang}" and "{target_lang}" are substituted by the source English and German respectively in the English to German experiments). This is an adaptation of the `GEMBA-DA` framework proposed in (Kocmi and Federmann, 2023).

[1] https://github.com/google-research/mt-metrics-eval.
[2] More information in https://platform.openai.com/docs/models

Table 3.1: Base prompt for the $k-$shot MT evaluation experiments. The empty lines are for aesthetic purpose only.

Prompt
Score the following translation from {source_lang} to {target_lang} on a continuous scale from 0 to 100, where a score of zero means "no meaning preserved" and a score of one hundred means "perfect meaning and grammar".Dont give any explanation. Take the following examples: {source_lang} source: "{src_example}$_1$" {target_lang} translation: "{mt_example}$_1$" Score given by human: {human_score_example}$_1$ ... {source_lang} source: "{src_example}$_k$" {target_lang} translation: "{mt_example}$_k$" Score given by human: {human_score_example}$_k$ Your turn: {source_lang} source: "{source_seg}" {target_lang} translation: "{target_seg}" Score:

The pool of examples provided for the few-shot scenario is obtained from a datastore that considers MQM annotated segments from the WMT 2019, 2020 and 2021 shared tasks, described as

$$(\mathcal{K}, \mathcal{V}) = \{(\boldsymbol{x}, \boldsymbol{y}) \mid (\boldsymbol{x}, \boldsymbol{y}) \in (\mathcal{X}_{\text{WMT}}, \mathcal{Y}_{\text{WMT}})\}, \tag{3.1}$$

where $(\mathcal{X}_{\text{WMT}}, \mathcal{Y}_{\text{WMT}})$ is the collection of parallel text from WMT general news task of 2019, 2020 and 2021.

The examples are fetched using three types of querying methods. Considering a k-shot scenario we tried querying the:

- **A)** k most similar sources and then choose a translation from a random system;
- **B)** k most similar concatenation of source and machine translation (if we have ten different systems, for each source segment we will have ten different concatenations of source and machine translation);
- **C)** most similar source and choose k most similar translations associated with that source.

This lets us analyse whether the fetched examples are having a substantial influence on the model's final output and whether this influence relies more on source or MT. By trying these different querying methods we can also analyse whether providing multiple examples associated with one single source helps the model to better "learn" the task rather than multiple different examples.

The results about system-wise accuracy (Kocmi et al., 2021) are shown on Table 3.2.

Table 3.2: Results of system level accuracy for English to German and Chinese to English language pairs, using the `GEMBA-DA` framework. In yellow are the zero-shot experiments made by (Kocmi and Federmann, 2023).

Model	Setup	Reference	Query Method	Accuracy (en-de)(%)	Accuracy (zh-en)(%)	Accuracy (%)
GPT-3.5	2 shot	No	A	**0.949**	**0.890**	**0.917**
GPT-3.5	5 shot	No	A	0.910	0.879	0.894
Davinci-003	0 shot	Yes	-	0.923	0.868	0.893
GPT-3.5	2 shot	No	B	0.936	0.853	0.890
GPT-3.5	2 shot	No	C	0.923	0.861	0.889
GPT-3.5	1 shot	No	B	0.935	0.851	0.888
GPT-4	0 shot	Yes	-	0.897	0.868	0.882
GPT-3.5	1 shot	No	A	0.904	0.861	0.881
GPT-3.5	2 shot	Yes	B	0.905	0.853	0.877
GPT-3.5	1 shot	No	C	0.891	0.862	0.876
GPT-3.5	1 shot	Yes	A	0.885	0.862	0.873
GPT-3.5	2 shot	Yes	A	0.885	0.858	0.871
GPT-3.5	1 shot	Yes	C	0.882	0.846	0.863
GPT-3.5	1 shot	Yes	B	0.876	0.841	0.857
Davinci-002	0 shot	Yes	-	0.872	0.835	0.852
GPT-3.5	2 shot	Yes	C	0.877	0.830	0.852
Davinci-002	0 shot	No	-	0.885	0.824	0.852
GPT-4	0 shot	No	-	0.846	0.857	0.852
Davinci-003	0 shot	No	-	0.872	0.824	0.846
COMET-22	-	Yes	-	0.769	0.868	0.822
GPT-3.5	0 shot	No	-	0.782	0.857	0.822
BLEURT-20	-	Yes	-	0.769	0.846	0.822
GPT-3.5	0 shot	Yes	-	0.795	0.780	0.787
COMET-QE-22	-	No	-	0.718	0.813	0.769

The first aspect to notice is the large increase in system-level accuracy of the few-shot experiments comparing to the zero-shot ones with the `gpt-3.5-turbo`. In fact, with few-shot learning, `gpt-3.5-turbo`, which was the worst model in the experiments made by (Kocmi and Federmann, 2023), ends up surpassing the best model for these two language pairs. This is a good indication that experimenting with the other models such as `text-davinci-003` or `gpt-4` would yield even stronger results, since both these models substantially outperformed `gpt-3.5-turbo` in the zero-shot scenario, and have a bigger model size (`gpt-4` is a direct improvement of the `gpt-3.5-turbo`). However, this is a large-sized dataset and these models are substantially more expensive, which is why no further experiments were made in this section.

The different query methods of the few-shot examples seem to have a slight effect on the performance of the `gpt-3.5-turbo`, although it is not clear. On the 2 shot experiments without reference, query method "A" seems to yield best results which is surprising due to this method being a more general approach in contrast with query method "B", since the pool of available examples from method "A" is a subset of the pool from method "B". However, in the remainder few-shot experiments, query method "B"

outperforms. The last query method seems to consistently yield the worst results. Another interesting trend is the fact that when using few-shot experiments the inclusion of the reference seems to deteriorate results which is a contrary trend to the one obtained in the zero-shot experiments.

Finally we can conclude that the trend of LLMs outperforming existent neural metrics such as reference and reference-free COMET-22 (Rei et al., 2022a) and BLEURT (Sellam et al., 2020) in system-level MT evaluation, maintains itself when using few-shot learning.

The results of the segment-level correlations of the same language pairs (English to German and Chinese to English) show substantially different trends and are shown in Tables 3.3 and 3.4.

Table 3.3: Segment-level correlations for English to German using the `GEMBA-DA` framework (Kocmi and Federmann, 2023). In yellow are the zero-shot experiments reported by the same paper.

Model	Setup	Reference	Query Method	Correlations (en-de) (Kendall's tau)
UniTE	-	Yes	-	**0.362**
COMET-22	-	Yes	-	0.361
MetricX-XXL	-	Yes	-	0.356
GPT-4	0 shot	Yes	-	0.347
BLEURT-20	-	Yes	-	0.338
GPT-4	0 shot	No	-	0.337
Davinci-003	0 shot	Yes	-	0.301
GPT-3.5	0 shot	Yes	-	0.299
COMET-QE	-	No	-	0.277
GPT-3.5	2 shot	No	A	0.266
GPT-3.5	2 shot	Yes	A	0.253
GPT-3.5	2 shot	No	B	0.241
GPT-3.5	1 shot	Yes	A	0.238
GPT-3.5	1 shot	Yes	C	0.233
GPT-3.5	2 shot	No	C	0.233
GPT-3.5	2 shot	Yes	B	0.232
GPT-3.5	1 shot	Yes	B	0.231
GPT-3.5	1 shot	No	A	0.230
GPT-3.5	2 shot	Yes	C	0.230
Davinci-002	0 shot	Yes	-	0.228
GPT-3.5	1 shot	No	C	0.228
GPT-3.5	1 shot	No	B	0.226
GPT-3.5	0 shot	No	-	0.225
GPT-3.5	5 shot	No	A	0.217
Davinci-002	0 shot	No	-	0.203
Davinci-003	0 shot	No	-	0.176

Table 3.4: Segment-level correlations for Chinese to English using the `GEMBA-DA` framework (Kocmi and Federmann, 2023). In yellow are the zero-shot experiments reported by the paper.

Model		Reference	**Query Method**	**Correlations (zh-en) (Kendall's tau)**
MetricX-XXL	-	Yes	-	**0.421**
COMET-22	-	Yes	-	0.420
GPT-4	0 shot	No	-	0.394
GPT-4	0 shot	Yes	-	0.370
Davinci-003	0 shot	Yes	-	0.360
COMET-QE	-	No	-	0.356
BLEURT-20	-	Yes	-	0.352
GPT-3.5	0 shot	No	-	0.352
UniTE	-	Yes	-	0.351
GPT-3.5	0 shot	Yes	-	0.344
Davinci-002	0 shot	Yes	-	0.294
Davinci-003	0 shot	No	-	0.275
GPT-3.5	2 shot	Yes	A	0.271
Davinci-002	0 shot	No	-	0.270
GPT-3.5	2 shot	Yes	C	0.269
GPT-3.5	2 shot	Yes	B	0.266
GPT-3.5	1 shot	Yes	A	0.253
GPT-3.5	2 shot	No	C	0.249
GPT-3.5	2 shot	No	A	0.248
GPT-3.5	1 shot	Yes	B	0.245
GPT-3.5	1 shot	No	A	0.244
GPT-3.5	1 shot	Yes	C	0.243
GPT-3.5	5 shot	No	A	0.238
GPT-3.5	1 shot	No	C	0.238
GPT-3.5	2 shot	No	B	0.236
GPT-3.5	1 shot	No	B	0.232

The results of the few-shot experiments for the segment-level experiments were completely outclassed by the zero-shot ones from (Kocmi and Federmann, 2023) for both English to German and Chinese to English by a large amount. Both of these LLMs-based metrics are outclassed by neural-based metrics. Only in the English to German language pair do the few-shot `gpt-3.5-turbo` reference-free experiments outclass the zero-shot experiment. On the other hand, GPT-4 is, by far, the best performing LLM, both with and without the use of reference, managing to compete with the state-of-the-art neural metrics, which begs the question of whether few-shot learning on top of this model would improve the results even further.

On the other hand, inside the few-shot learning experiments we see an inverse trend to what was observed in system-level, which is that the reference-based metrics outperform the reference-free metrics.

Another relevant aspect to analyse is the reliability of the answers obtained, which is affected by the model's temperature value. The temperature is a tweakable parameter that affects the output provided by LLMs, by changing the probability distribution of the generated output. Decoding with higher

temperature leads to greater linguistic variety, while a low value tends to output more correct and deterministic sentences, with low variability. Consequently, in research papers temperature is usually fixed to zero, for example, (Peng et al., 2023) found that generally a lower setting of temperature generally results in a higher performance for MT using *in-context learning*. In (Kocmi and Federmann, 2023), each time an invalid answer was obtained (such as a textual answer or an invalid number) the experiment was repeated with a slight increase of temperature.

We perform the exact same method of increasing temperature upon invalid answer, however it is interesting to note that with the model `gpt-3.5-turbo` they obtained 565 and 935 invalid answers for the reference-based and reference-free experiments, while with few-shot we obtained 0 and 3 respectively. This indicates that providing few-shot examples helps the model better comply with the task-specific rules.

(Kocmi and Federmann, 2023) also noted that the DA reference-based model outputs mostly scores that are multiples of 5 and that over three-quarters are either 80, 95 or 100. On the other hand, for the reference-free experiments 60.5% of scores outputted were the number 95. We computed histograms of the scores given by human judgements, the best model from the zero-shot experiments (0 shot `text-davinci-003`), and our 2 shot reference-based and reference-free experiments, shown in Figure 3.1.

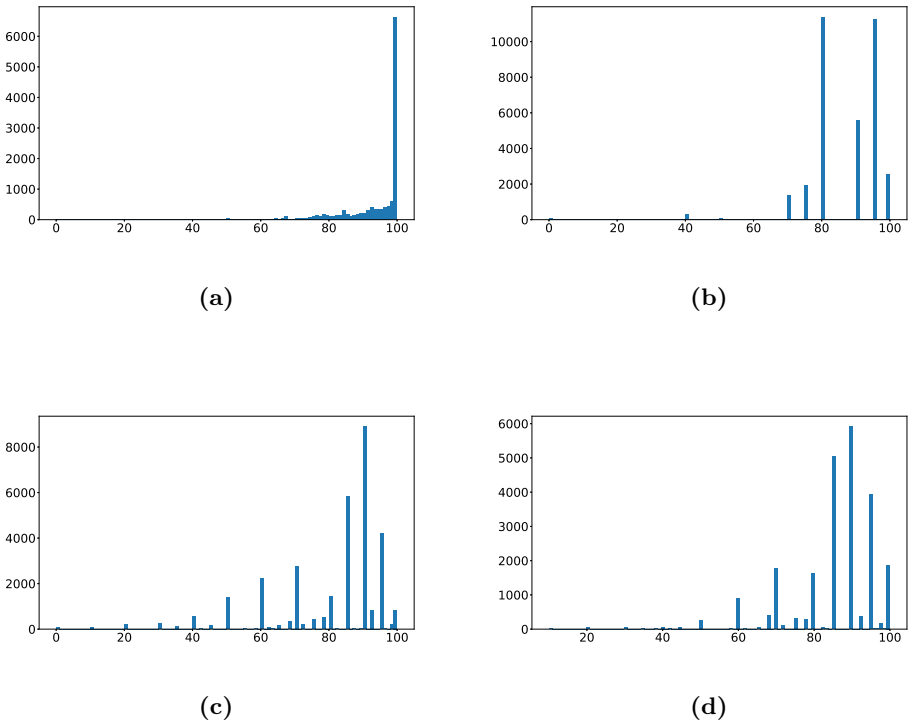


Figure 3.1: DA scores obtained by (a) human judgements, (b) `text-davinci-003` 0 shot reference-based experiment (c) `gpt-3.5-turbo` 2 shot reference-free experiment and (d) `gpt-3.5-turbo` 2 shot reference-based experiment.

When adding few-shot examples, the LLM outputs a much more distributed score histogram, which can be the reason as to why our models obtained such low segment-level correlations but good system-level accuracy. The model is correctly ranking systems among each other, but is using a much more distributed score board which causes a substantial decrease in segment-level correlations. The histograms obtained for the remainder of the made experiments yielded similar distributions to plots 3.1(c) and 3.1(d).

3.2.1 Experiments with Random Scores

(Min et al., 2022) concluded that randomly replacing the demonstrations labels of an *in-context* classification experiment using LLMs barely hurts the model's performance. We perform a similar experiment to analyse the effect of the labels of the in-context examples on the final score outputted by the LLM. We replace the human scores of the datastore contents with a random score and analyse model's performance. The results are shown in Tables 3.5 and 3.6. The query method used was "B" since there was not significative performance difference between methods "A" and "B", and the pool of possible examples provided by method"A" is a subset of of the pool provided by method "B".

Table 3.5: System-wise accuracy of using in-context examples with random wrong scores assigned for English to German.

Model	Setup	Reference	Query Method	Sys-Acc [en-de] [%]
GPT-3.5	2 shot	No	B	0.949
GPT3.5	1 shot	No	B	0.936
GPT-3.5	2 shot-Wrong	No	B	0.923
GPT-3.5	5 shot-Wrong	No	B	0.923
GPT-3.5	1 shot-Wrong	No	B	0.916
GPT-3.5	5 shot	No	B	0.910

Table 3.6: Segment-level Kendall correlations when using in-context examples with random wrong scores assigned for English to German.

Model	Setup	Reference	Query Method	Correlations [en-de] (Kendall's tau)
GPT-3.5	2 shot	No	B	0.241
GPT-3.5	1 shot	No	B	0.227
GPT-3.5	5 shot	No	B	0.222
GPT-3.5	5 shot-Wrong	No	B	0.138
GPT-3.5	2 shot-Wrong	No	B	0.135
GPT-3.5	1 shot-Wrong	No	B	0.133

Even though the prompt describes this as a regression problem, the fact is that `gpt-3.5-turbo` treats this as a classification problem, mostly outputting scores that are multiples of 5. Table 3.6 shows that, contrarily to findings made by (Min et al., 2022) for a classification task, providing wrong labels substantially hurts segment-level scores. However, as observed in 3.5 the model is still able to correctly rank systems among each other system-wise. This behaviour maintains when considering reference-based

experiments as well as the Chinese to English language pair. The histogram of DA scores of the 1 shot experiment with random scores is shown in Figure 3.2 (the 2 and 3 shot experiments provide similar histograms).

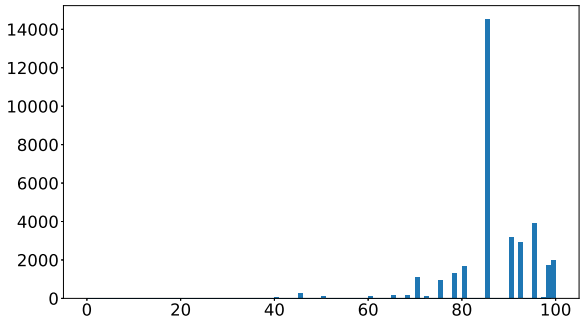


Figure 3.2: Histogram of DA scores provided by `gpt-3.5-turbo` 1 shot random scores experiment.

Among all experiments these were the ones where the LLM outputted the same score the most, which could be a sign of random guessing because of the wrong in-context examples. We can deduce that the in-context examples actually provide information relevant to the final score outputted by the LLM, which is an indicator that better and more diverse in-context examples might provide a better final segment-level correlation.

3.3 Main Findings

GPT-based LLMs perform well when assessing the quality of machine translation on a system-level, which means that given translated documents by different systems, the model is able to, for the most part, correctly rank them up based on their performance. However, when considering the most general use-case of assessing the quality of individual segments, these models severely underperform existent state-of-the-art neural metrics, which were fine-tuned to perform on a segment-level. For the system-level experiments using `gpt-3.5-turbo`, the few-shot scenario substantially improves the zero-shot one while on the segment-level it actually ends up deteriorating the results. The models, especially in a zero-shot scenario, tend to mostly output the same scores (scenario alleviated when providing few-shot examples), which could be an indicator of random guessing. However, when considering the same setups with random DA scores, the model tends to aggravate this situation of mostly outputting the same score, leading to considerably worse performance at the segment-level. On a system-level, even with a low performance at segment-level when providing examples with wrong labels, the model is still able to correctly rank systems among each other.

4

Terminology-Constrained Machine Translation

Contents

Despite Neural Machine Translation (NMT) achieving remarkable progress in translation quality, there remains consistent shortcomings in some of its applications for different use cases and content types. One of such shortcomings is correctly translating domain-specific terminology for some particular content types.

In this section we will evaluate and analyse recent widely used LLMs for terminology-constrained MT, through the use of *in-context learning* ability that some LLMs possess, using zero and few-shot learning approaches fetching examples from a local datastore. In this Chapter we compare some of our work with a previous approach for terminology-constrained MT (Dinu et al., 2019), where the authors train a MT model to handle terminology constraints during inference. Previous approaches focused on constrained decoding, an approximate search algorithm capable of enforcing any constraints over resulting output sequences. This introduces a substantial computational overhead in the decoding phase during inference and shows inflexibility and stiffness when including terminology.

Experimental Setup

The datasets used are portions of the English to German publicly available terminology databases, Wiktionary and IATE[1], which is constitutes around 727 sentences for the Wiktionary test set and 414 sentences for the IATE one. Furthermore, terminology entries that occur in the English top 500 most frequent words, or that are single character were removed, as was done by (Dinu et al., 2019). Finally, the term bases were divided in two different sets, training and test, making sure there is no overlap on the source side, similarly to (Dinu et al., 2019).

The experiments were made using the `gpt3.5-turbo`[2] model with a cutoff date of June 2023. All embeddings used as keys of the datastores for the *in-context learning* experiments were computed using the LaBSE model (Feng et al., 2022) and fetched using the euclidean distance. To create the datastores and perform the nearest neighbour search, the library FAISS (Johnson et al., 2019) was used. General quality measurements are obtained using the metrics COMET, BLEU and chrF.

4.1 Zero-Shot Scenario

The base prompt that was used is shown in Table 4.1, where elements within curly brackets are replaced by the correspondent experiment specific information (as an example, "{source language}" is replaced by "english" in the English to German experiments).

[1] More information in https://iate.europa.eu and https://www.wiktionary.org/
[2] https://platform.openai.com/docs/models.

Table 4.1: Base version of the zero-shot scenario prompt for the task of terminology-constrained Machine Translation (MT).

Prompt
Translate the following sentence from English to German without providing any explanation and using the provided glossary. Glossary: {terminology$_1$ in source_language}={terminology$_1$ in target_language} ; ... ; {terminology$_k$ in source_language}={terminology$_k$ in target_language} {source_language} source: {source_sentence}. Your {target_language} translation:

In order to evaluate the effect that the prompt has on these experiments, a simpler variation of this prompt without an initial description of the task, shown in Table 4.2, was also experimented.

Table 4.2: Simpler variation of the base terminology-constrained machine translation prompt.

Prompt
Glossary: {terminology$_1$ in source_language}={terminology$_1$ in target_language} ; ... ; {terminology$_k$ in source_language}={terminology$_k$ in target_language} {source_language} source: {source_sentence}. Your {target_language} translation:

4.2 Few-Shot Scenario

4.2.1 Datastore with terminology-constrained examples

The first attempt at a few-shot learning scenario was to consider a datastore using terminology-constrained examples. As explained earlier, we divided the terminology dataset into two parts, training and test, to mimic the experimental scenario done in (Dinu et al., 2019). Consequently, and since we are not training an MT model, we can use the training set as content of a datastore. The correspondent datastore is formalised as

$$(\mathcal{K}, \mathcal{V}) = \{(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \mid (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \in (\mathcal{X}_{wikt}, \mathcal{Y}_{wikt}, \mathcal{T}_{wikt})\}, \tag{4.1}$$

where $(\mathcal{X}_{wikt}, \mathcal{Y}_{wikt}, \mathcal{T}_{wikt})$ is the collection of Wiktionary training data source, reference and terminology respectively, for each language pair. Similarly we create a datastore for the IATE training set described as,

$$(\mathcal{K}_2, \mathcal{V}_2) = \{(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \mid (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{t}) \in (\mathcal{X}_{iate}, \mathcal{Y}_{iate}, \mathcal{T}_{iate})\}. \tag{4.2}$$

The k-shot learning scenario base prompt, as well as its simpler version are shown in Tables 4.3 and 4.4 respectively.

Table 4.3: Base terminology-constrained machine translation k-shot scenario prompt.

Prompt
Translate the following sentence from {source_language} to {target_language} without providing any explanation and using the provided glossary. Example 1: Glossary: {terminology$_{(1,1)}$ in source_language}={terminology$_{(1,1)}$ in target_language} ; ... ; {terminology$_{(1,n_1)}$ in source_language}={terminology$_{(1,n_1)}$ in target_language} {source_language} source: {source_sentence}. {target_language} translation: {reference_translation} ... Example k: Glossary: {terminology$_{(k,1)}$ in source_language}={terminology$_{(k,1)}$ in target_language} ; ... ; {terminology$_{(k,n_k)}$ in source_language}={terminology$_{(k,n_k)}$ in target_language} {source_language} source: {source_sentence}. {target_language} translation: {reference_translation} Your turn: Glossary: {terminology$_{(k+1,1)}$ in source_language}={terminology$_{(k+1,1)}$ in target_language} ; ... ; {terminology$_{(k+1,n_{k+1})}$ in source_language}={terminology$_{(k+1,n_{k+1})}$ in target_language} {source_language} source: {source_sentence}. Your {target_language} translation:

Table 4.4: Simpler variation of the base terminology-constrained machine translation k-shot scenario prompt.

Prompt
Example 1: Glossary: {terminology$_{(1,1)}$ in source_language}={terminology$_{(1,1)}$ in target_language} ; ... ; {terminology$_{(1,n_1)}$ in source_language}={terminology$_{(1,n_1)}$ in target_language} {source_language} source: {source_sentence}. {target_language} translation: {reference_translation} ... Example k: Glossary: {terminology$_{(k,1)}$ in source_language}={terminology$_{(k,1)}$ in target_language} ; ... ; {terminology$_{(k,n_k)}$ in source_language}={terminology$_{(k,n_k)}$ in target_language} {source_language} source: {source_sentence}. {target_language} translation: {reference_translation} Your turn: Glossary: {terminology$_{(k+1,1)}$ in source_language}={terminology$_{(k+1,1)}$ in target_language} ; ... ; {terminology$_{(k+1,n_{k+1})}$ in source_language}={terminology$_{(k+1,n_{k+1})}$ in target_language} {source_language} source: {source_sentence}. Your {target_language} translation:

Results and analysis

The main objective of these experiments is to analyse the ability of Large Language Model (LLM) to accurately and consistently make use of the proper terminology, while maintaining or improving the translation quality. For this purpose we consider the percentage of times the term translation was generated in the output out of the total number of term annotations (identified as the term percentage), as well as widely used quality metrics such as BLEU, COMET, and chrF.

Results are shown in Tables 4.5 and 4.6 for both tested versions of the prompt using the test set without terminology constraint (baseline), and with terminology constraint considering k examples ($k \in \{0, 1, 2, 3\}$). The prompt that yielded the baseline simply instructs the LLM to translate the sentences from English to German.

Table 4.5: Term percentage and quality scores (BLEU, COMET and chrF) of terminology-constrained MT for the English to German language pair using the base prompt.

Wiktionary Base prompt, English-German				
	Term Percentage (%)	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	87.75	55.46	87.50	63.85
0 shot	95.80	56.39 (+0.93)	87.72 (+0.22)	64.67 (+0.82)
1 shot	96.15	56.17 (+0.71)	87.77 (+0.27)	64.45 (+0.60)
2 shot	96.03	56.53 (+1.07)	87.88 (+0.38)	64.76 (+0.91)
3 shot	96.50	56.53 (+1.07)	**87.90 (+0.40)**	64.70 (+0.85)
4 shot	**96.58**	**56.74 (+1.28)**	**87.90 (+0.40)**	**64.79 (+0.94)**

IATE Base prompt, English-German				
	Term Percentage (%)	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	84.27	54.85	87.05	62.96
0 shot	95.80	55.66 (+0.81)	87.40 (+0.35)	63.85 (+0.89)
1 shot	95.73	55.66 (+0.81)	87.43 (+0.38)	63.94 (+0.98)
2 shot	**95.96**	55.41 (+0.56)	87.44 (+0.39)	63.86 (+0.90)
3 shot	95.28	55.88 (+1.03)	87.52 (+0.47)	64.13 (+1.17)
4 shot	95.93	**55.97 (+1.12)**	**87.54 (+0.49)**	**64.23 (+1.27)**

Table 4.6: Term percentage and quality scores (BLEU, COMET and chrF) of terminology-constrained MT for the English to German language pair using the simpler prompt.

Wiktionary Base prompt, English-German				
	Term Percentage (%)	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	87.75	55.46	87.50	63.85
0 shot	95.45	53.10 (-2.36)	85.65 (-1.85)	64.07 (+0.22)
1 shot	**96.50**	56.49 (+1.03)	87.93 (+0.43)	64.82 (+0.97)
2 shot	96.27	56.65 (+1.19)	88.01 (+0.51)	65.02 (+1.17)
3 shot	96.38	**56.66 (+1.20)**	88.02 (+0.52)	65.03 (+1.18)
4 shot	**96.50**	**56.66 (+1.20)**	**88.16 (+0.66)**	**65.07 (+1.22)**

IATE Base prompt, English-German				
	Term Percentage (%)	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	84.27	54.85	87.05	62.96
0 shot	94.38	51.66 (-3.19)	84.34 (-2.71)	63.01 (+0.05)
1 shot	95.73	55.20 (+0.35)	87.32 (+0.27)	63.59 (+0.63)
2 shot	95.73	55.56 (+0.71)	87.42 (+0.37)	63.90 (+0.94)
3 shot	95.73	**56.08 (+1.23)**	87.63 (+0.58)	64.38 (+1.42)
4 shot	**96.88**	56.01 (+1.16)	**87.69 (+0.64)**	**64.59 (+1.63)**

Analysing Table 4.5, term percentages largely increase when the glossary is introduced (which corresponds to the 0 to 4 shot scenarios). Actually, the 3 to 5 % of terms where the LLM is not able to output the correct terminology corresponds to 29 to 34 sentences in the Wiktionary dataset and 18 to 21 sentences in the IATE dataset (depending on the experiment), for each k shot experiment ($k \in \{0, 1, 2, 3\}$). In fact, due to the small size of the test set and the high term percentages, we can manually analyse the instances where the model failed to include the requested terms. Such examples are shown in Appendix A.

The first situation in which the model fails is using synonyms, or even different words, of the ones in the presented terminology, as in the examples shown in Tables A.1 and A.2, where the model uses the word "Abend" instead of "Nacht", or in Table A.4 where the model uses its own expression "beiden Seiten" instead of the requested "beidseitig", or in Table A.3 where the model substitutes the word "league" ("Liga") with the synonym "class" ("Klasse"). Secondly we have situations in which the model ignores the glossary completely, such as in Table A.5 where the model uses the full word "Weltmeisterschaft" instead of the requested abbreviation "WM". Finally we have situations in which the models uses the requested word but inflected, or uses the word with a different casing than the requested as in the example shown in Table A.6, in which the model uses "Die republikanischen" instead of the requested "Die Republikaner", or in Table A.7 in which the model uses "statt" instead of the terminology "stattfinden".

It is also worthy to note that more than half ($> 70\%$) of the sentences in which the model failed to use the correct terminology, on any given experiment of the IATE dataset, are common to all other experiments. This makes sense since the term percentage improvements when adding more examples to the prompt are not considerable. On the Wiktionary dataset this value is similar to the one in the IATE dataset.

Still on the base prompt experiments, comparing the 0 shot experiment with the baseline, all quality scores improve substantially. COMET is a neural metric which was expected to improve in this experiment since it considers both the semantics and lexicality of a translation against its reference. Older lexical-based MT metrics such BLEU and chrF also improved a great amount which indicates that the LLM is able to insert the requested terminology while adapting surroundings words, contrary to previously existent methods such as constrained decoding (Chatterjee et al., 2017, Hokamp and Liu, 2017, Hasler et al., 2018). In addition, quality scores increase a slight amount the more examples added to the prompt, with the majority of the best results appearing int the 4 shot experiment.

Looking at the experiments with the simpler version of the prompt, shown in Table 4.6, the term percentage parameter is quite similar among both tables, with differences of 1 to 2 terminology errors when comparing to the other prompt. The quality scores however have a different pattern. First of all the 0 shot experiment substantially decreases BLEU and COMET scores while chrF scores slightly increase on the Wikitionary dataset and slightly decrease on the IATE dataset. This means that providing no task description and no examples severely deteriorates results. On the other hand, when providing in-context examples (1 to 3 shot scenarios), quality scores greatly improve over the baseline. In fact, these improvements are larger than the experiments using the base prompt in about 60%, 25% and 30% (relative BLEU, COMET and chrF)for the Wiktionary experiments and 30%, 25% and 10% for the IATE experiments.

Finally, we can compare the results from both tables with the experiments performed by (Dinu et al., 2019). The term percentages obtaining by any of our k-shot learning experiments were substantially higher than both the methods proposed by (Dinu et al., 2019). In addition we obtained significantly higher quality scores improvements across all metrics used. Only the constrained decoding (Hokamp and Liu, 2017) method can obtain higher term percentage values with a sufficiently large beam size, with the costs of lower quality scores and a drastic latency increase.

4.2.2 Datastore With Large Amounts of Parallel Data

The extremely small size of the datastore in the previous experiment (168 sentences for the IATE experiments and 248 sentences for the Wiktionary experiments) is a bottleneck. Since the retrieval is done by source similarity, having a larger sized datastore allows to retrieve better examples. Following this, a different style of datastore was experimented using the training data of the WMT 2017 news translation

task (which contains around 6M news-related sentences), without the terminology component, described as

$$(\mathcal{K}, \mathcal{V}) = \{(\boldsymbol{x}, \boldsymbol{y}) \mid (\boldsymbol{x}, \boldsymbol{y}) \in (\mathcal{X}_{17}, \mathcal{Y}_{17})\}, \tag{4.3}$$

where $(\mathcal{X}_{17}, \mathcal{Y}_{17})$ is the collection of parallel training data from the WMT-17 news translation task.

Through this experiment we can analyse the trade-off between providing better source-translation examples but without the in-context terminology component or less similar source-translation examples but with the terminology component (experiments presented in the previous section). The results are shown in Table 4.7.

Table 4.7: Term percentage and quality scores (BLEU, COMET and chrF) of terminology-constrained MT for the English to German language pair using the base prompt and a large datastore with WMT-17 training data.

Wiktionary **Base prompt, English-German**				
	Term Percentage (%)	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	87.75	55.46	87.50	63.85
0 shot	**95.80**	56.39 (+0.93)	87.72 (+0.22)	64.67 (+0.82)
1 shot	95.33	**56.61 (+1.15)**	**87.85 (+0.35)**	64.69 (+0.84)
2 shot	95.57	56.49 (+1.03)	87.81 (+0.31)	64.79 (+0.78)
3 shot	95.45	56.54 (+1.08)	87.79 (+0.29)	**64.85 (+1.00)**
4 shot	95.22	56.63 (+1.09)	87.84 (+0.34)	64.75 (+0.90)

IATE **Base prompt, English-German**				
	Term Percentage (%)	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	84.27	54.85	87.05	62.96
0 shot	**95.80**	55.66 (+0.81)	**87.40 (+0.35)**	63.85 (+0.89)
1 shot	95.28	**56.09 (+1.24)**	87.33 (+0.28)	**64.03 (+1.07)**
2 shot	95.06	55.89 (+1.04)	87.28 (+0.23)	63.92 (+0.96)
3 shot	95.51	56.02 (+1.17)	87.31 (+0.26)	63.90 (+0.94)
4 shot	95.83	55.77 (+0.92)	87.39 (+0.34)	63.77 (+0.81)

The first main difference from the experiments of the previous section is that now the best results appear mostly in the $0-1$ shot experiments (for both quality scores and term percentages). This indicates that by providing more out-of-context examples does not help the model to include terminology, which is to be expected, but also does not help to increase quality scores, which is surprising as we are now providing better examples from a larger pool of sentences.

The errors obtained in the terminology inclusion of these experiments are of the same types as the ones in the previous section, with the vast majority of these errors appearing in the same sentences as in

last experiment. Which is expected on considering that the values of term percentage are quite similar to the ones obtained in the previous section.

4.2.3 Computational Overhead

The 99^{th} percentile latency for our experiments as well as the ones made by (Dinu et al., 2019), are shown in Table 4.8.

Table 4.8: 99^{th} percentile latency for constrained decoding, methods proposed in (Dinu et al., 2019) and LLM-based approach.

Model	Time (s)
Constrained Decoding	0.68
Train by appending	0.19
Train by replacing	0.20

Model	Time (s)
Api call	2.25

Model	Time (s)	
	Datastore with task-specific examples	Datastore with large amounts of parallel data
GPT-3.5 1-shot	0.01	3.99
GPT-3.5 2-shot	0.02	4.00
GPT-3.5 3-shot	0.02	3.99
GPT-3.5 4-shot	0.02	4.00

The computational overhead of introducing few-shot learning to OpenAI models, such as `gpt-3.5-turbo` and `gpt-4`, is minimal if using a very small-sized datastore (less than 300 sentences), like the one used in subsection 4.2.1 and considering scenarios with similar prompt and examples sizes as int he experiments made. However, when we consider a larger datastore, as the one used in 4.2.2 with around 6M sentences, the computational overhead increases by 200 to 400 times.

It is also important to note that, for closed LLMs, there is a time associated to making the API call to send and receive messages to the LLMs. These constraints are imposed by OpenAI as a response for handling a high number of API calls every day. This, in combination with the retrieval of few-shot examples for large-sized datastores, is a bottleneck in applications that require high throughput, where open-sourced LLMs or other MT systems should be used.

4.3 Main Findings

Just by providing a glossary to a normal machine translation-like prompt (0 shot scenario) the term percentage metric goes up a large amount when comparing it to the baseline. In fact, the term percentage obtained by the zero-shot scenario is considerably larger than one obtained by the method proposed by (Hokamp and Liu, 2017) (about 4% higher on their best translation quality model and about 2% on their best term percentage model). Only the constrained decoding method obtains higher terminology

percentage values at the cost of lowering the quality scores and a drastic latency increase. When we consider the few-shot examples provided by a very small-sized datastore, we get a slight increase in terminology percentage with the addition of substantially increasing all three measured quality scores. Due to the small-sized datastore, this method provided very little extra computational overhead on top of using the zero-shot LLM, despite still not comparing with the proposed methods by (Dinu et al., 2019) due to the time it takes to make the API call.

Overall `gpt-3.5-turbo` performs well in terminology-constrained translation, especially when considering previous methods, with a substantial increase of terminology percentage and quality scores provided by the inclusion of few-shot examples. Changing the prompt can influence the results, in these experiments we considered a simpler version of the prompt without a task description which led to greater results in every scenario where a example of the task was provided. Without both the example and the task description, the model performs a substantially poorer in terms of quality score, and slightly worse regarding terminology inclusion. Finally, the use of a large-sized datastore with translation examples, without the terminology inclusion portion of the task performs worse in every aspect.

5

Automatic Translation Post-Editing

Contents

APE refers to the task of proposing improvements over a given translation, T, and generating the translation with the proposed improvements T^+.

Although state-of-the-art neural machine translation models has greatly improved over the past few years (Wang et al., 2022), machine translation models are still far from being reliable, even for high-resource language pairs (He et al., 2020, Gupta et al., 2020, Wang et al., 2021, He et al., 2021, Raunak et al., 2022). In addition, the rise of popularity of NMT systems also reinforces the importance of having high quality post-editing systems. Recent work has been done on investigating the ability of LLMs (such as GPT 3.5 and 4) to handle the task of automatically post-editing of machine translations, due to their versatility and recent popularity, on a zero-shot scenario (Raunak et al., 2023). This work focuses on the nature of the post-edited translation, general quality improvements, edits on human annotated error spans and fidelity of proposed edits. This chapter proposes to further research this work, including an analysis of the capability of these LLMs to automatically choose which machine translations benefit from a post-edit step as well as extending all existent research to a few-shot scenario.

Experimental Setup

The datasets used are the WMT 2022 general machine translation task (Kocmi et al., 2022) and WMT-21 news translation task annotated with MQM errors (Freitag et al., 2021), namely the German to English and the Ukrainian to Czech language pairs, with the first one being a high resource language pair and the second one being a low resource language pair. For the first language pair the best and worst systems were used, `Lan-Bridge` and `PROMT` respectively, and for the second language pair `ALMAnaCH-Inria`.

The post-editing experiments were made using `gpt3.5-turbo` and GPT-4[1], both with cutoff dates in June of 2023, *i.e.*, the model has not received any updates since the referred date. All embeddings used as keys of the datastores for the in-context learning experiments were computed using the LaBSE model (Feng et al., 2022) and fetched using the euclidean distance. To create the datastores and perform the nearest neighbour search, the FAISS library (Johnson et al., 2019) was used. General quality measurements are obtained using the metrics COMET, BLEU and chrF.

5.1 Zero-Shot Scenario

The base prompt used is in Table 5.1, where elements within curly brackets are substituted by the correspondent experiment specific information (as an example, "{source language}" is substituted by "german" in the German to English experiments).

[1] https://platform.openai.com/docs/models/gpt-4.

Table 5.1: Base post-editing zero-shot scenario prompt.

Prompt
You're going to improve a given sentence which is a machine translation in {target_language} from a source sentence in {source_language}, without providing explanations. {source_language} source: {source_sentence} {target_language} machine translation: {translation} Your improved translation (in {target_language}):

5.2 Few-Shot Scenario

5.2.1 Datastore With Parallel Data

The first attempt at a few-shot scenario was to consider a datastore that included WMT-22's German to English training set, composed of nearly $35M$ sentences from publicly available data sources, such as the Europarl corpus, the UN corpus, the news commentary corpus and the ParaCrawl corpus[2]. The correspondent datastore is formalised as

$$(\mathcal{K}, \mathcal{V}) = \{(\boldsymbol{x}, \boldsymbol{y}) \mid (\boldsymbol{x}, \boldsymbol{y}) \in (\mathcal{X}_{22}, \mathcal{Y}_{22})\}, \tag{5.1}$$

where $(\mathcal{X}_{22}, \mathcal{Y}_{22})$ is the WMT-22 training data collection of parallel text (sentences in the source language, $\mathcal{X}_{22}$, and their translation, $\mathcal{Y}_{22}$), for each language pair.

The k-shot scenario base prompt is shown in Table 5.2.

Table 5.2: Base post-editing k-shot scenario prompt.

Prompt
You're going to improve a given sentence which is a machine translation in {target_language}from a source sentence in {source_language}, without providing explanations and considering examples of similar great translations. Example 1: {source_language} source: {source_sentence}. {target_language} translation: {reference_translation} ... Example k: {source_language} source: {source_sentence}. {target_language} translation: {reference_translation} Your turn: {source_language} source: {source_sentence}. {target_language} machine translation: {translation}. Your improved translation (in {target_language}):

[2]More information regarding the used datasets is available on the official WMT-22 website: https://www.statmt.org/wmt22/translation-task.html.

Results and analysis

The results obtained are shown in Table 5.3, where we used the winning system of WMT-22 English to German general translation shared task, `Lan-Bridge`. The baseline of each experiment corresponds to the system's performance on the translation task without any post-editing. In the 0-4 shot scenarios a post-editing step is applied, using the prompt shown in Table 5.1, with k examples for *in-context learning*, where $k \in \{0, 1, 2, 3, 4\}$.

Table 5.3: Results of automatic post-editing experiments for the `Lan-Bridge` system and the German to English language pair using `gpt-3.5-turbo`.

Datastore size: 3.4M sentences Base prompt, German-English, `gpt-3.5-turbo` Sys: `Lan-Bridge`,			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**56.89**	**85.63**	**59.64**
0 shot	51.81 (-5.05)	61.91 (-23.72)	55.51 (-4.98)
1 shot	51.88 (-5.01)	62.00 (-23.63)	55.73 (-3.91)
2 shot	51.84 (-5.05)	62.07 (-23.56)	55.64 (-4.00)
3 shot	51.48 (-5.41)	62.01 (-23.62)	55.42 (-4.22)
4 shot	51.31 (-5.58)	62.23 (-23.40)	55.18 (-4.46)

Datastore size: 17M sentences Base prompt, German-English, `gpt-3.5-turbo` Sys: `Lan-Bridge`,			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**56.89**	**85.63**	**59.64**
0 shot	51.81 (-5.05)	61.91 (-23.72)	55.51 (-4.98)
1 shot	51.50 (-5.39)	62.08 (-23.55)	55.64 (-4.00)
2 shot	51.47 (-5.42)	62.08 (-23.55)	55.60 (-4.04)
3 shot	51.05 (-5.84)	62.11 (-23.52)	55.25 (-4.39)
4 shot	51.05 (-5.84)	62.09 (-23.54)	55.22 (-4.42)

Datastore size: 34M sentences Base prompt, German-English, `gpt-3.5-turbo` Sys: `Lan-Bridge`			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**56.89**	**85.63**	**59.64**
0 shot	51.81 (-5.05)	61.91 (-23.72)	55.51 (-4.98)
1 shot	51.83 (-5.03)	62.09 (-23.54)	55.79 (-3.85)
2 shot	51.52 (-5.37)	62.06 (-23.57)	55.63 (-4.01)
3 shot	51.13 (-5.76)	62.06 (-23.57)	55.29 (-4.35)
4 shot	50.96 (-5.93)	62.05 (-23.58)	55.46 (-4.18)

The most important aspects to notice from the results are the fact that not only does providing more *in-context learning* examples leads to a quality decrease over the zero-shot scenario, applying post-editing

at all decreases every quality score a large amount when considering the baseline. One characteristic of this experiment, which can help to explain this failure, is the fact the used system, `Lan-Bridge`, is the winning translator for WMT-22 general MT task, producing the best machine translations. One hypothesis is that `gpt3.5-turbo` is not able to distinguish good translations from bad ones, creating bad modifications on good translations which deteriorate their quality. In fact, by further investigating the LLM's responses we can observe this behaviour, as seen in the example shown in Table 5.4, where, despite the APE system being given a perfect translation, because of its nature and the prompt, it still tries to modify it resulting in a worse output ("The item cost less than 20 euros."). This happens several times in these experiments.

Table 5.4: Post-editing zero-shot scenario example.

Prompt	You're going to improve a given sentence which is a machine translation in English from a source sentence in German, without providing explanations and considering examples of similar great translations. German source: Die Ware hat unter 20 Euro gekostet. English machine translation: **The goods cost less than 20 euros.** Your improved translation (in english):
Answer	The item cost less than 20 euros.
Reference	**The goods cost less than 20 euros.**

To reinforce this hypothesis, Figure 5.1 shows a plot of the difference between scores obtained in the baseline and the APE COMET scores for the experiments using the datastore with 3.4M sentences. The experiments using different sized datastores show similar behaviour.

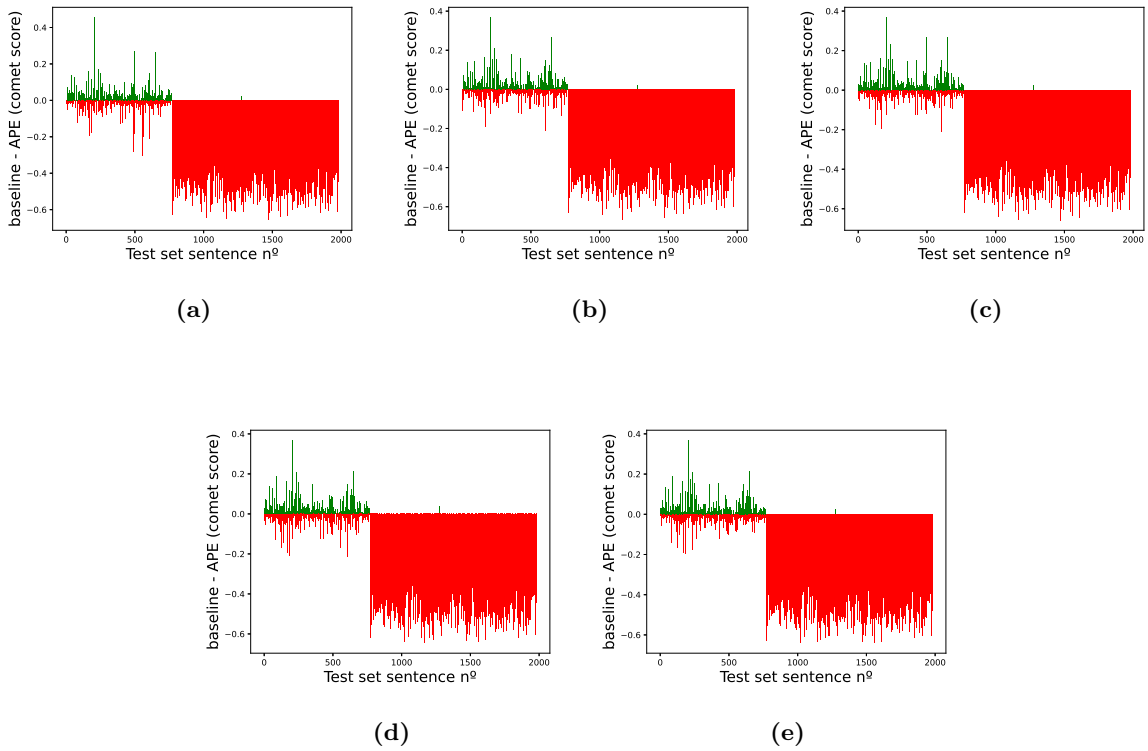


Figure 5.1: Changes in COMET of the (a) 0, (b) 1, (c) 2, (d) 3 and (e) 4 shot APE experiments against the baseline for the 3.4M sized datastore.

As we can see from Figure 5.1, COMET scores after post-editing drop in $81-82$ % of the test set, for all experiments, which confirms the previous hypothesis. In addition, if we consider the last two thirds of the test set, the scores drop in more than 99% of sentences, and the COMET scores drop by a large amount. This is because translations in the last two thirds of the dataset all have high COMET scores (greater than 90%), contrarily to the first third of the dataset where the quality scores are considerably lower. This behaviour maintains when considering the few-shot learning scenario for the other datastore sizes (17M sentences and 35M sentences).

It is also important to emphasize that this difficulty of `gpt-3.5-turbo` in differentiating good from bad translations is not surprising, since, in fact, APE systems do not usually have this ability. What has happenned is the pairing of a APE system with a QE system that detects and prevents quality translations from a post-edit step (Chatterjee et al., 2018).

In order to tackle this behaviour, a different experiment was performed where the post-editing is only applied to sentences where the COMET score of the baseline translation is bellow a certain threshold, similarly to the work of (Chatterjee et al., 2018). This simulates the practical post-editing scenario present in machine translation pipelines shown in Figure 5.2.

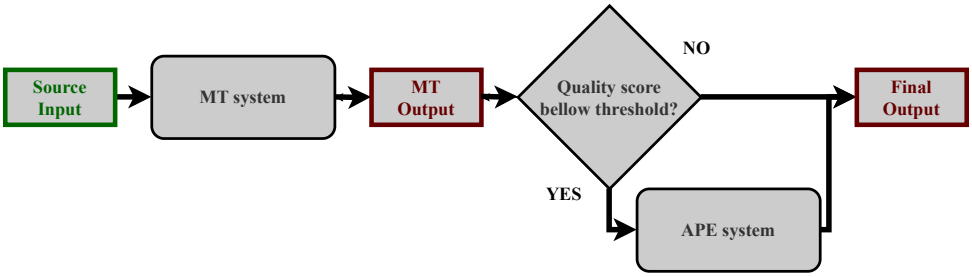


Figure 5.2: Pipeline of a practical post-editing scenario that makes it so only poor translations are directed to the APE system.

The results for the experiments with the scenario described above with a COMET-QE score threshold of 85%, 80% and 65% respectively, which were obtained with the `wmt22-cometkiwi-da` model, are shown in Tables 5.5, 5.6 and 5.7. The baseline also only considers the respective subset of machine translations, which is why it changes from each of Tables 5.5, 5.6 and 5.7.

Table 5.5: Results of automatic post-editing experiments for the German to English language pair using the base prompt, considering segments with COMET-QE score inferior to 85, for the `Lan-Bridge` system.

Datastore size: 3.4M sentences **Base prompt, German-English, `gpt-3.5-turbo`** **Sys: `Lan-Bridge`, Only COMET-QE < 85**			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**49.63**	78.97	**52.21**
0 shot	47.92 (-1.71)	80.25 (+1.28)	51.61 (-0.60)
1 shot	47.64 (-1.99)	80.43 (+1.46)	51.63 (-0.58)
2 shot	47.80 (-1.83)	80.36 (+1.39)	51.85 (-0.36)
3 shot	47.95 (-1.68)	80.44 (+1.47)	51.90 (-0.31)
4 shot	47.83 (-1.80)	**80.50 (+1.53)**	51.75 (-0.46)

Datastore size: 17M sentences **Base prompt, German-English, `gpt-3.5-turbo`** **Sys: `Lan-Bridge`, Only COMET-QE < 85**			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**49.63**	78.97	**52.21**
0 shot	47.92 (-1.71)	80.25 (+1.28)	51.61 (-0.60)
1 shot	47.96 (-1.67)	80.50 (+1.53)	51.94 (-0.27)
2 shot	47.75 (-1.88)	**80.53 (+1.56)**	51.80 (-0.41)
3 shot	47.64 (-1.99)	80.48 (+1.51)	51.66 (-0.55)
4 shot	47.83 (-1.80)	80.51 (+1.54)	51.69 (-0.51)

Datastore size: 34M sentences Base prompt, German-English, `gpt-3.5-turbo` Sys: `Lan-Bridge`, Only COMET< 85			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**49.63**	78.97	**52.21**
0 shot	47.92 (-1.71)	80.25 (+1.28)	51.61 (-0.60)
1 shot	48.20 (-1.43)	80.40 (+1.43)	52.09 (-0.12)
2 shot	48.07 (-1.56)	80.43 (+1.46)	51.98 (-0.23)
3 shot	47.84 (-1.79)	80.40 (+1.43)	51.90 (-0.31)
4 shot	47.72 (-1.91)	**80.59 (+1.62)**	51.65 (-0.56)

Table 5.6: Results of automatic post-editing experiments for the German to English language pair using the base prompt, considering segments with COMET-QE score inferior to 80, for the `Lan-Bridge` system.

Datastore size: 3.4M sentences Base prompt, German-English, `gpt-3.5-turbo` Sys: `Lan-Bridge`, Only COMET-QE < 80			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**46.06**	73.90	48.28
0 shot	45.11 (-0.95)	76.59 (+2.69)	48.63 (+0.35)
1 shot	45.40 (-0.66)	76.97 (+3.07)	48.89 (+0.61)
2 shot	45.43 (-0.63)	76.91 (+3.01)	**49.13 (+0.85)**
3 shot	45.33 (-0.73)	76.87 (+2.97)	49.00 (+0.72)
4 shot	45.18 (-0.88)	**77.00 (+3.10)**	48.70 (+0.42)

Datastore size: 17M sentences Base prompt, German-English, `gpt-3.5-turbo` Sys: `Lan-Bridge`, Only COMET-QE < 80			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**46.06**	73.90	48.28
0 shot	45.11 (-0.95)	76.59 (+2.69)	48.63 (+0.35)
1 shot	45.46 (-0.60)	76.91 (+3.01)	48.89 (+0.61)
2 shot	45.11 (-0.95)	**77.12 (+3.22)**	48.81 (+0.53)
3 shot	44.89 (-1.17)	76.93 (+3.03)	48.63 (+0.35)
4 shot	45.59 (-0.47)	76.97 (+3.07)	**49.04 (+0.76)**

Datastore size: 34M sentences Base prompt, German-English, `gpt-3.5-turbo` Sys: `Lan-Bridge`, Only COMET-QE < 80			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**46.06**	73.90	48.28
0 shot	45.11 (-0.95)	76.59 (+2.69)	48.63 (+0.35)
1 shot	46.03 (-0.03)	**76.89 (+2.99)**	**49.39 (+1.11)**
2 shot	45.68 (-0.38)	76.70 (+2.80)	48.97 (+0.69)
3 shot	45.29 (-0.77)	76.87 (+2.97)	48.92 (+0.66)
4 shot	45.36 (-0.70)	76.83 (+2.93)	48.80 (+0.52)

Table 5.7: Results of automatic post-editing experiments for the German to English language pair using the base prompt, considering segments with COMET-QE score inferior to 65, for the `Lan-Bridge` system.

Datastore size: 3.4M sentences Base prompt, German-English, `gpt-3.5-turbo` Sys: `Lan-Bridge`, Only COMET-QE< 65			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	25.07	54.12	28.43
0 shot	33.06 (+7.99)	64.75 (+10.63)	34.75 (+6.32)
1 shot	**36.55 (+11.48)**	**67.14 (+13.02)**	**37.92 (9.49)**
2 shot	35.30 (+10.23)	66.70 (+12.68)	35.57 (+7.14)
3 shot	35.21 (+10.14)	65.53 (+11.41)	34.77 (+6.34)
4 shot	34.68 (+9.61)	65.80 (+11.68)	34.23 (+5.80)

Datastore size: 17M sentences Base prompt, German-English, `gpt-3.5-turbo` Sys: `Lan-Bridge`, Only COMET-QE < 65			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	25.07	54.12	28.43
0 shot	**33.06 (+7.99)**	64.75 (+10.63)	34.75 (+6.32)
1 shot	32.63 (+7.56)	65.23 (+11.11)	34.62 (+6.19)
2 shot	33.01 (+7.94)	**66.67 (+12.55)**	**35.08 (+6.65)**
3 shot	31.34 (+6.27)	63.44 (+9.32)	32.33 (+3.90)
4 shot	32.80 (+7.73)	63.69 (+9.57)	32.20 (+3.77)

Datastore size: 34M sentences Base prompt, German-English, `gpt-3.5-turbo` Sys: `Lan-Bridge`, Only COMET-QE < 65			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	25.07	54.12	28.43
0 shot	33.06 (+7.99)	64.75 (+10.63)	34.75 (+6.32)
1 shot	**33.31 (+8.24)**	**67.37 (+13.25)**	**36.06 (+7.63)**
2 shot	32.45 (+7.38)	67.19 (+13.07)	35.04 (+6.61)
3 shot	32.05 (+6.98)	63.55 (+9.43)	34.47 (+6.04)
4 shot	31.21 (6.14)	65.57 (+11.45)	34.38 (+5.95)

These experiments confirm the fact that `gpt-3.5-turbo` is not able to deduce on its own which translations would benefit from a post-editing step and which are not able to do so. Since from Tables 5.5-5.7 we are considering translations with a lower quality score, it is relevant to analyse the difference in quality scores regarding the baseline (marked with the Δ symbol). Every quality score increases strictly and considerably from each Table. Just by considering translations with worse COMET-QE score than 85%, which is already quite high, we already see substantial improvements in COMET score.

As a general trend for the three experiments, adding few-shot *in-context learning* improves all quality scores over the zero-shot scenario. However, when we compare the results against the baseline only the COMET scores increase consistently. In Table 5.5 only COMET scores improve, in Table 5.6 both

COMET and chrF improve and in Table 5.7 all quality scores improve over the baseline. Finally, analysing the datastore size variation, the datastore with the largest size achieves the best quality scores in the most cases, in the experiments where the few-shot scenario improves over the baseline, which is intuitive since the experiment is repeated with a larger pool of examples to retrieve. Despite this, the changes are not very substantial when considering the magnitude of the improvements.

5.2.2 Quality Indication (QI) On The Prompt

In the previous section it was established that LLMs are not able to select which machine translations are worthy of performing post-editing. In order to further research on this statement we will experiment an alternative approach with the objective of assessing the model's selection capabilities by changing the wording on the prompt to include a quality indication and instruct the model to only make this selection if necessary. The new prompt is shown in Table 5.8, while the results of the correspondent experiments are shown in Table 5.9.

Table 5.8: Updated base post-editing k-shot scenario prompt with inclusion of a quality indication component, which is shown highlighted in bold.

Prompt	You're going to improve a given sentence **(only if an improvement is possible)** which is a machine translation in {target language} from a source sentence in {source language}, without providing explanations and considering examples of similar great translations. **If an improvement is not possible output the same machine translation.** Example 1: {source language} source: {source sentence}. {target language} translation: {reference translation} ... Example k: {source language} source: {source sentence}. {target language} translation: {reference translation} Your turn: {source language} source: {source sentence}. {target language} machine translation: {translation}. **Quality score: {bad/ok/excellent}** Your improved translation (in {target language}):

The provided quality indication is obtained by first calculating the COMET-QE score of the machine translation, using the reference-free model, `wmt22-cometkiwi-da`. This is because we want to make the scenario as realistic as possible, and having references during a post-editing step is not possible. After computing the QE score they are distributed by the three possible labels, {`excellent, ok, bad`} through manually applied thresholds (scores around the baseline's mean were given the `ok` tag, substantially above scores were given the `excellent` tag and substantially lower scores were given the `bad`).

Table 5.9: Results of automatic post-editing experiments for the German to English language pair using a modification of the base prompt that includes a quality indication.

Datastore size: 3.4M sentences **Base prompt, German-English, `gpt-3.5-turbo`** **Sys: `Lan-Bridge`, QI on prompt**			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**56.89**	**85.63**	**59.64**
0 shot	52.61 (-4.28)	85.29 (-0.34)	56.34 (-3.30)
1 shot	52.63 (-4.26)	85.28 (-0.35)	56.35 (-3.29)
2 shot	52.33 (-4.56)	85.28 (-0.35)	56.22 (-3.42)
3 shot	52.13 (-4.76)	85.28 (-0.35)	56.07 (-3.57)
4 shot	51.69 (-4.20)	85.19 (-0.44)	55.74 (-3.90)

Datastore size: 17M sentences **Base prompt, German-English, `gpt-3.5-turbo`** **Sys: `Lan-Bridge`, QI on prompt**			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**56.89**	**85.63**	**59.64**
0 shot	**52.61 (-4.28)**	85.29 (-0.34)	56.34 (-3.30)
1 shot	52.83 (-4.06)	85.36 (-0.27)	56.76 (-2.88)
2 shot	52.52 (-4.37)	85.38 (-0.25)	56.40 (-3.24)
3 shot	52.07 (-4.82)	85.28 (-0.35)	55.99 (-3.64)
4 shot	52. 14 (-4.75)	85.22 (-0.41)	56.02 (-3.62)

Datastore size: 34M sentences **Base prompt, German-English, `gpt-3.5-turbo`** **Sys: `Lan-Bridge`, QI on prompt**			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**56.89**	**85.63**	**59.64**
0 shot	52.61 (-4.28)	85.29 (-0.34)	56.34 (-3.30)
1 shot	52.90 (-3.99)	85.31 (-0.32)	56.62 (-3.02)
2 shot	52.33 (-4.56)	85.26 (-0.37)	56.13 (-3.51)
3 shot	52.12 (-4.77)	85.20 (-0.43)	56.01 (-3.63)
4 shot	51.92 (-4.97)	85.22 (-0.41)	55.78 (-3.86)

Analysing the results, we observe that the model is able to adapt really well to the new information of a quality indication indicated in the prompt with high quality. The scores are greatly superior to the vanilla case with the base prompt, despite still not being able to surpass the baseline. *i.e.*, the model's performance without an post-editing step. This indicates that the model's weakness of not being able to distinguish poor from great translations can be substantially mitigated through a change in prompt. In these experiments, adding more than one few-shot example deteriorates the results, since the best setup occurs mostly in the $1-2$ shot scenario for all metrics. Consequently, the effect of the datastores size is also not noticeable.

5.2.3 Worse Machine Translation System

Earlier in this chapter it was stated one of the reasons `gpt-3.5-turbo` struggled in the task of APE was because the experiments were done on the winning system of the German to English WMT-22 general machine translation task, `Lan-Bridge` (Kocmi et al., 2022), which produced a substantial amount of translations that do not benefit from a post-editing step.

The next natural step is to consider the worst performing system on this translation task, which is the `PROMT` system. The results are shown in Table 5.10.

Table 5.10: Results of automatic post-editing experiments for the German to English experiments using the base prompt on the worst performing system of WMT-22 for this language pair, `PROMT`.

Datastore size: 3.4M sentences Base prompt, German-English, `gpt-3.5-turboo` Sys: `PROMT`			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**53.90**	**83.84**	**57.04**
0 shot	53.00 (-0.90)	65.52 (-18.32)	57.01 (-0.03)
1 shot	52.63 (-1.27)	65.46 (-18.38)	56.64 (-0.40)
2 shot	52.91 (-0.99)	65.50 (-18.34)	56.89 (-0.15)
3 shot	52.45 (-1.45)	65.50 (-18.34)	56.70 (-0.34)
4 shot	52.43 (-1.47)	65.49 (-18.35)	56.66 (-0.38)

Datastore size: 17M sentences Base prompt, German-English, `gpt-3.5-turbo` Sys: `PROMT`			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**53.90**	**83.84**	57.04
0 shot	53.00 (-0.90)	65.52 (-18.32)	57.01 (-0.03)
1 shot	53.15 (-0.75)	65.61 (-18.23)	**57.21 (+0.17)**
2 shot	52.98 (-0.92)	65.57 (-18.27)	57.17 (+0.13)
3 shot	52.68 (-1.22)	65.59 (-18.25)	56.91 (-0.13)
4 shot	52.37 (-1.53)	65.56 (-18.28)	56.63 (-0.41)

Datastore size: 34M sentences Base prompt, German-English, `gpt-3.5-turbo` Sys: `PROMT`			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**53.90**	**83.84**	57.04
0 shot	53.00 (-0.90)	65.52 (-18.32)	57.01 (-0.03)
1 shot	53.20 (-0.70)	65.57 (-18.27)	**57.11 (+0.07)**
2 shot	53.08 (-0.82)	65.55 (-18.29)	57.08 (+0.04)
3 shot	52.67 (-1.23)	65.53 (-18.31)	56.89 (-0.15)
4 shot	52.43 (-1.47)	65.50 (-18.34)	56.66 (-0.38)

The results are positive compared to the initial experiments with the winning system of WMT-22 (in

Table 5.3). The system is able to actually obtain substantially higher values of COMET, BLEU and chrF scores across all experiments, and is able to outperform the baseline considering the chrF score for the last two experiments with datastore sizes of 17M and 34M sentences. These improvements indicate that there are less situations with a good translation hypothesis which is where the model struggles. In fact if we re-analyse the example in Table 5.4, it still happens in these experiments, which means that the core problem persists. The model is simply able to perform better due to worse machine translations at its input. This reinforces the previous conclusions.

5.2.4 Experiments Using GPT-4

In this subsection we analyse the performance of a better LLM model which is the GPT-4[3], a direct improvement over the `gpt-3.5-turbo` model. Mainly, it is relevant to analyse the ability for GPT-4 to detect which translations are already of high quality and do not benefit from a post-editing step. The results are shown in Table 5.11.

Table 5.11: Results of automatic post-editing German to English experiments for the base prompt using GPT-4.

Datastore size: 34M sentences Base prompt, German-English, GPT-4 Sys: `Lan-Bridge`			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**56.89**	**85.63**	**59.64**
0 shot	49.89 (-7.00)	85.56 (-0.07)	54.66 (-4.98)
1 shot	48.21 (-8.68)	85.23 (-0.4)	53.81 (-5.83)
2 shot	49.14 (-7.75)	85.35 (-0.28)	53.88 (-5.76)
3 shot	48.94 (-7.95)	85.53 (-0.10)	53.81 (-5.83)
4 shot	48.90 (-7.99)	85.30 (-0.33)	53.64 (-6.00)

Datastore size: 34M sentences Base prompt, German-English, GPT-4 Sys: `Lan-Bridge`			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**56.89**	**85.63**	**59.64**
0 shot	49.89 (-7.00)	85.56 (-0.07)	54.66 (-4.98)
1 shot	48.22 (-8.67)	85.30 (-0.33)	53.30 (-6.34)
2 shot	49.21 (-7.68)	85.37 (-0.26)	54.02 (-5.62)
3 shot	49.20 (-7.69)	85.36 (-0.27)	53.95 (-5.69)
4 shot	49.26 (-7.63)	85.36 (-0.27)	53.97 (-5.67)

[3] https://platform.openai.com/docs/models/gpt-4.

Datastore size: 34M sentences Base prompt, German-English, GPT-4 Sys: `Lan-Bridge`			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	**56.89**	**85.63**	**59.64**
0 shot	49.89 (-7.00)	85.56 (-0.07)	54.66 (-4.98)
1 shot	47.19 (-9.70)	84.91 (-0.72)	52.22 (-7.42)
2 shot	48.12 (-8.77)	84.97 (-0.66)	52.84 (-6.80)
3 shot	48.15 (-8.79)	84.98 (-0.65)	52.79 (-6.85)
4 shot	48.26 (-8.63)	85.00 (-0.63)	52.86 (-6.78)

Comparing Table 5.3 with Table 5.11, GPT-4 significantly outperforms gpt-3.5-turbo when considering COMET scores, which demonstrates its superior ability in distinguishing good translations from poor ones. However, it is important to note that the lexical-based quality metrics, BLEU and chrF, are significantly lower than the ones obtained through `gpt-3.5-turbo`. On the other hand, the inclusion of few-shot examples for the *in-context learning* task seems to considerably deteriorate all quality scores when comparing to the zero-shot scenario. Figure B.1 indicates that GPT-4 is better at distinguishing good from bad translations than `gpt-3.5-turbo`.

5.2.5 Low-resource Language Pair

So far only high-resource language pairs were considered. In this subsection we test the ability of the higher performant APE system,GPT-4, in the same scenario as the previous experiments but in a low-resource language pair, which is Ukrainian to Czech. The results are shown in Table 5.12.

Table 5.12: Results of automatic post-editing experiments for the Ukrainian to Czech language pair using the base prompt and using the GPT-4 model, for the system `ALMAnaCH-Inria`.

Datastore size: 3.4M sentences Base prompt, Ukranian-Czech, gpt-4 Sys: `ALMAnaCH-Inria`			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	43.61	82.30	49.80
0 shot	47.04 (+3.43)	85.23 (+2.93)	53.13 (+3.33)
1 shot	51.16 (+7.55)	88.72 (+6.42)	55.86 (+6.06)
2 shot	51.67 (+8.06)	89.91 (+7.31)	57.12 (+7.32)
3 shot	51.93 (+8.32)	90.59 (+8.29)	57.58 (+7.78)
4 shot	**52.05 (+8.44)**	**91.34 (+9.04)**	**58.44 (+8.64)**

Datastore size: 17M sentences Base prompt, Ukranian-Czech, gpt-4 Sys: `ALMAnaCH-Inria`			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	43.61	82.30	49.80
0 shot	47.04 (+3.43)	85.23 (+2.93)	53.13 (+3.33)
1 shot	51.11 (+7.50)	90.89 (+8.59)	57.96 (8.16)
2 shot	52.18 (+8.57)	91.51 (+9.21)	58.42 (+8.62)
3 shot	52.20 (+8.59)	91.70 (+9.40)	58.71 (+8.91)
4 shot	**52.42 (+8.81)**	**91.81 (+9.51)**	**58.89 (+9.09)**

Datastore size: 34M sentences Base prompt, Ukranian-Czech, gpt-4 Sys: `ALMAnaCH-Inria`			
	BLEU (Δ)	COMET (Δ)	chrF (Δ)
baseline	43.61	82.30	49.80
0 shot	47.04 (+3.43)	85.23 (+2.93)	53.13 (+3.33)
1 shot	50.27 (+6.66)	89.58 (+7.28)	55.76 (+5.96)
2 shot	51.91 (+8.30)	91.52 (+9.22)	58.19 (+8.39)
3 shot	52.36 (+8.85)	91.79 (+9.49)	**58.74 (+8.94)**
4 shot	**52.48 (+8.87)**	**92.13 (+9.83)**	58.67 (+8.87)

Analysing the results, GPT-4 is able to perform very well in this low-resource language pair. All quality metrics increase over the baseline by a large amount, and also, the more few-shot examples are provided in the prompt, the higher the quality scores for the most cases, although the biggest increases occurs from the 0 to 1 shot and from 1 to 2 shot experiments. The best results out of the three Tables were obtained in the datastore with the pool of 34M sentences for BLEU and COMET, while chrF was higher in the datastore with 17M sentences (by a small amount).

5.3 Main Findings

LLMs are able to successfully perform in the task of automatically post-editing a machine translation through *in-context learning*. They are not able, however, to automatically identify which translations are already good enough to not require a post-editing step. The worse the translations provided to the tested GPT-based language models, the better are the quality improvements obtained. The model GPT-4 is a substantially better APE system than `gpt-3.5-turbo`, which is to be expected since the first model is a direct improvement over the second one.

The versatility shown by LLMs is also shown and reinforced here through the fact that a single model is able to perform post-editing automatically and for multiple languages, whether they are high-resource or low-resource languages, in different types of contexts and without any further training.

6

Conclusion

Contents

This thesis focused on leveraging the technique of *in-context learning* for recently developed widely-used LLMs, such as `gpt-3.5-turbo` and GPT-4.

6.1 Conclusions and Achievements

LLMs have achieved ground-breaking development with the introduction of dozens of new models in the past few years, as shown in Figure 2.13. The power to extend and diversify their range of skills with the increase of model size and training data, known as the emergent skills (Wei et al., 2022), is one of their main reasons for interest. The ability to perform well on different tasks without further training, in addition with the ease of access to some of these LLMs, is a very desirable feature on numerous different areas, one of them being MT. This thesis shows the versatility of LLMs by testing them on different MT applications through the the emergent ability of *in-context learning*.

In Chapter 3 were tested the few-shot capabilities of `gpt-3.5-turbo` on MT evaluation against the zero-shot experiments with `gpt-3.5-turbo` and GPT-4, as well as state-of-the-art neural metrics, with and without the use of references. Although the segment-level results obtained by GPT LLMs are still not comparable to current state-of-the-art neural metrics, the document-level results achieve state-of-the-art results. With the use of few-shot examples, the document-level pair-wise accuracy substantially increases over the zero-shot experiments using `gpt-3.5-turbo`, even surpassing the results obtained by other better performing LLM models such as `davinci-003` and `gpt-4`, achieving new state-of-the-art results. On the other hand, at the segment-level the tendency is different, with a clear over-performance by the neural metrics such as UniTE and COMET. When comparing the results against the zero-shot experiments, adding the few-shot examples only deteriorates the results in every experiment except in the English to German language pair, where reference-free few-shot experiments improve segment-level correlations over zero-shot scenario with `gpt-3.5-turbo`.

In Chapter 4 the focus changed to terminology constrained MT, where `gpt-3.5-turbo` showed great improvements not only regarding term percentage of terminology, but also in translation quality. However, upon inspection of the few failure scenarios, it was clear that the LLM tends to sometimes ignore the instruction completely, using term synonyms for examples, which raises reliability issues. It was also shown that for this task, it is better to include worse examples that are task-specific than than to provide examples with more similar general machine translations.

Lastly, in Chapter 5 the challenge is performing automatic post-editing of machine translations. The tested LLMs show poor capabilities of distinguishing bad translations from great ones that do not benefit from a post-editing step, although `gpt-4` considerably outperforms `gpt-3.5-turbo` in this matter. However, when provided with only poor translations, which is the realistic situation in a real-world APE pipeline, the model is able to perform the correct modifications needed to improve the quality scores. A big advantage of LLMs is their ability to adapt to diverse scenarios, which was demonstrated through

different experiments that varied the prompt template according to certain assumptions.

As a final note, this thesis implicitly reinforces the amazing versatility that LLMs have. It was shown that a single LLM could perform every step of a real-world MT pipeline, ranging from machine translation, MT evaluation and automatic post-editing, using *in-context learning*.

6.2 Future Work

6.2.1 Considering Different LLMs

As introduced in Chapter 2, the world of LLMs is rapidly evolving and there are several relevant models with different characteristics than the models used. In this thesis, GPT models were exclusively used due to computation and cost constraints, as well as their ease of use when comparing to other models. As a next step, this research could be extend to other undisclosed models, such as BARD and PaLM (Chowdhery et al., 2022), or even open-sourced models such as LLaMa2 (Touvron et al., 2023b). One advantage of using open-sourced LLMs is we have access to the underlying probability distributions. Currently, through few-shot learning we can only influence these probabilities indirectly, through the prompt, however, if we could change the distributions, we could perform more fine-grained types of in-context learning, for example retrieving words instead of sentences, and also perform a retrieval step at each word generation step, similar to what was done by (Khandelwal et al., 2020), on retrieval-based language modelling.

Furthermore, in the MT evaluation segment-level experiments, few-shot GPT-3.5 had poor performance when comparing to neural metrics and the other LLMs, although improving on its zero-shot scenario. Consequently, it would be interesting to analyse how these other, more expensive, LLMs would fair when given few-shot examples, especially GPT-4, which as a zero-shot scenario yielded competitive results to the neural metrics. It would be interesting to analyse if the few-shot scenario using GPT-4 would outclass the best neural metrics for the segment-level.

6.2.2 Fine-Tuning

This thesis focused on exploring the *in-context learning* ability of LLMs that arise when scaling the model size and training data. Another relevant route which was impossible in this thesis scope due to cost and computation constraints, is to analyse performance of these models fine-tuned for MT tasks, and compare these results to the performance obtained through in-context learning. Fine-tuning is the task adjusting the model's parameters to perform better on specific tasks, which usually involves further training the model on a smaller and task-targeted dataset. However, this comes at an additional cost, especially for open-source models, where, because of the model sizes, this task involves a great amount of computation power.

6.2.3 More and Better Datasets

In all experiments publicly available datasets were used, which was a bottleneck in some cases. In the terminology constrained machine translation experiments, there were very few publicly datasets available, and several of the research made in this area used private datasets. Moreover, this data was very small-sized and included only one language pair. Consequently the datastore that included task-related examples was very small, which limited the scope of the experiments. It would be interesting to consider a greater amount of small domain-specific datasets with different levels of terminology variety and specificity.

On the other hand, the task specific datasets used for Chapter 3 were also small-sized which constituted a bottleneck for analysing the in-context learning technique.

6.2.4 Different Tasks

It is clear that the versatility of LLMs are revolutionising the world in many tasks. In this thesis we focus only on analysing few-shot *in-context learning* in three MT relevant scenarios, however, there are a lot more tasks that benefit from a similar study, such as classification, math, code generation, interaction with the world, interaction with humans, among many others (Bubeck et al., 2023).

Bibliography

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bai et al., 2022] Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., Chen, C., Olsson, C., Olah, C., Hernandez, D., Drain, D., Ganguli, D., Li, D., Tran-Johnson, E., Perez, E., Kerr, J., Mueller, J., Ladish, J., Landau, J., Ndousse, K., Lukosuite, K., Lovitt, L., Sellitto, M., Elhage, N., Schiefer, N., Mercado, N., DasSarma, N., Lasenby, R., Larson, R., Ringer, S., Johnston, S., Kravec, S., Showk, S. E., Fort, S., Lanham, T., Telleen-Lawton, T., Conerly, T., Henighan, T., Hume, T., Bowman, S. R., Hatfield-Dodds, Z., Mann, B., Amodei, D., Joseph, N., McCandlish, S., Brown, T., and Kaplan, J. (2022). Constitutional ai: Harmlessness from ai feedback.

[Banerjee and Lavie, 2005] Banerjee, S. and Lavie, A. (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*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

[Bengio et al., 2000] Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. volume 3, pages 932–938.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):11371155.

[Blatz et al., 2004] Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A., and Ueffing, N. (2004). Confidence estimation for machine translation. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 315–321, Geneva, Switzerland. COLING.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):9931022.

[Bojar et al., 2015] Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Huck, M., Hokamp, C., Koehn, P., Logacheva, V., Monz, C., Negri, M., Post, M., Scarton, C., Specia, L., and Turchi, M. (2015). Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

[Bubeck et al., 2023] Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., and Zhang, Y. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4.

[Callison-Burch et al., 2011] Callison-Burch, C., Koehn, P., Monz, C., and Zaidan, O. (2011). Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, Scotland. Association for Computational Linguistics.

[Chatterjee et al., 2018] Chatterjee, R., Negri, M., Turchi, M., Blain, F., and Specia, L. (2018). Combining quality estimation and automatic post-editing to enhance machine translation output. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 26–38, Boston, MA. Association for Machine Translation in the Americas.

[Chatterjee et al., 2017] Chatterjee, R., Negri, M., Turchi, M., Federico, M., Specia, L., and Blain, F. (2017). Guiding neural machine translation decoding with external knowledge. In *Proceedings of the Second Conference on Machine Translation*, pages 157–168, Copenhagen, Denmark. Association for Computational Linguistics.

[Chen and Goodman, 1999] Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(4):359394.

[Cho et al., 2014a] Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder-decoder approaches.

[Cho et al., 2014b] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

[Chowdhery et al., 2022] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. (2022). Palm: Scaling language modeling with pathways.

[Chung et al., 2022] Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. (2022). Scaling instruction-finetuned language models.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160167, New York, NY, USA. Association for Computing Machinery.

[Correia and Martins, 2019] Correia, G. M. and Martins, A. F. T. (2019). A simple and effective approach to automatic post-editing with transfer learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3050–3056, Florence, Italy. Association for Computational Linguistics.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

[Dinu et al., 2019] Dinu, G., Mathur, P., Federico, M., and Al-Onaizan, Y. (2019). Training neural machine translation to apply terminology constraints.

[Dyer et al., 2013] Dyer, C., Chahuneau, V., and Smith, N. A. (2013). A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

[Elman, 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

[Feng et al., 2022] Feng, F., Yang, Y., Cer, D., Arivazhagan, N., and Wang, W. (2022). Language-agnostic bert sentence embedding.

[Fomicheva et al., 2020] Fomicheva, M., Sun, S., Yankovskaya, L., Blain, F., Chaudhary, V., Fishel, M., Guzmán, F., and Specia, L. (2020). BERGAMOT-LATTE submissions for the WMT20 quality estimation shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1010–1017, Online. Association for Computational Linguistics.

[Freitag et al., 2021] Freitag, M., Foster, G., Grangier, D., Ratnakar, V., Tan, Q., and Macherey, W. (2021). Experts, Errors, and Context: A Large-Scale Study of Human Evaluation for Machine Translation. *Transactions of the Association for Computational Linguistics*, 9:1460–1474.

[Freitag et al., 2022] Freitag, M., Rei, R., Mathur, N., Lo, C.-k., Stewart, C., Avramidis, E., Kocmi, T., Foster, G., Lavie, A., and Martins, A. F. T. (2022). Results of WMT22 metrics shared task: Stop using BLEU – neural metrics are better and more robust. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 46–68, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

[Graves and Schmidhuber, 2005] Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052 vol. 4.

[Gupta et al., 2020] Gupta, S., He, P., Meister, C., and Su, Z. (2020). Machine translation testing via pathological invariance. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, page 863875, New York, NY, USA. Association for Computing Machinery.

[Hasler et al., 2018] Hasler, E., de Gispert, A., Iglesias, G., and Byrne, B. (2018). Neural machine translation decoding with terminology constraints. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 506–512, New Orleans, Louisiana. Association for Computational Linguistics.

[Hauser et al., 2010] Hauser, M. D., Chomsky, N., and Fitch, W. T. (2010). *The faculty of language: what is it, who has it, and how did it evolve?*, page 1442. Approaches to the Evolution of Language. Cambridge University Press.

[He et al., 2020] He, P., Meister, C., and Su, Z. (2020). Structure-invariant testing for machine translation. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 961973, New York, NY, USA. Association for Computing Machinery.

[He et al., 2021] He, P., Meister, C., and Su, Z. (2021). Testing machine translation via referential transparency. In *Proceedings of the 43rd International Conference on Software Engineering*, ICSE '21, page 410422. IEEE Press.

[Heo et al., 2022] Heo, D., Lee, W., and Lee, J.-H. (2022). Quality estimation of machine translation using dual-encoder architecture. *Journal of KIISE*, 49:521–529.

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

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

[Hokamp and Liu, 2017] Hokamp, C. and Liu, Q. (2017). Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

[Huang et al., 2023] Huang, X., Ruan, W., Huang, W., Jin, G., Dong, Y., Wu, C., Bensalem, S., Mu, R., Qi, Y., Zhao, X., Cai, K., Zhang, Y., Wu, S., Xu, P., Wu, D., Freitas, A., and Mustafa, M. A. (2023). A survey of safety and trustworthiness of large language models through the lens of verification and validation.

[Johnson et al., 2019] Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

[Johnson et al., 2017] Johnson, J., Douze, M., and Jégou, H. (2017). Billion-scale similarity search with gpus.

[Junczys-Dowmunt and Grundkiewicz, 2016] Junczys-Dowmunt, M. and Grundkiewicz, R. (2016). Log-linear combinations of monolingual and bilingual neural machine translation models for automatic post-editing. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 751–758, Berlin, Germany. Association for Computational Linguistics.

[Junczys-Dowmunt and Grundkiewicz, 2018] Junczys-Dowmunt, M. and Grundkiewicz, R. (2018). MS-UEdin submission to the WMT2018 APE shared task: Dual-source transformer for automatic post-editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 822–826, Belgium, Brussels. Association for Computational Linguistics.

[Jégou et al., 2011] Jégou, H., Douze, M., and Schmid, C. (2011). Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33:117–28.

[Kalchbrenner and Blunsom, 2013] Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Conference on Empirical Methods in Natural Language Processing*.

[Kaplan et al., 2020] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models.

[Kasai et al., 2020] Kasai, J., Pappas, N., Peng, H., Cross, J., and Smith, N. A. (2020). Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation.

[Katz, 1987] Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401.

[Kendall, 1938] Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, 30(12):8193.

[Kepler et al., 2019] Kepler, F., Trénous, J., Treviso, M., Vera, M., and Martins, A. F. T. (2019). OpenKiwi: An open source framework for quality estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 117–122, Florence, Italy. Association for Computational Linguistics.

[Khandelwal et al., 2021] Khandelwal, U., Fan, A., Jurafsky, D., Zettlemoyer, L., and Lewis, M. (2021). Nearest neighbor machine translation. In *International Conference on Learning Representations*.

[Khandelwal et al., 2020] Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., and Lewis, M. (2020). Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

[Kim et al., 2017] Kim, H., Jung, H.-Y., Kwon, H., Lee, J.-H., and Na, S.-H. (2017). Predictor-estimator: Neural quality estimation based on target word prediction for machine translation. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 17:1–22.

[Kim and Lee, 2016] Kim, H. and Lee, J.-H. (2016). A recurrent neural networks approach for estimating the quality of machine translation output. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 494–498, San Diego, California. Association for Computational Linguistics.

[Kocmi et al., 2022] Kocmi, T., Bawden, R., Bojar, O., Dvorkovich, A., Federmann, C., Fishel, M., Gowda, T., Graham, Y., Grundkiewicz, R., Haddow, B., Knowles, R., Koehn, P., Monz, C., Morishita, M., Nagata, M., Nakazawa, T., Novák, M., Popel, M., and Popović, M. (2022). Findings of the 2022 conference on machine translation (WMT22). In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 1–45, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

[Kocmi and Federmann, 2023] Kocmi, T. and Federmann, C. (2023). Large language models are state-of-the-art evaluators of translation quality.

[Kocmi et al., 2021] Kocmi, T., Federmann, C., Grundkiewicz, R., Junczys-Dowmunt, M., Matsushita, H., and Menezes, A. (2021). To ship or not to ship: An extensive evaluation of automatic metrics for machine translation. In *Proceedings of the Sixth Conference on Machine Translation*, pages 478–494, Online. Association for Computational Linguistics.

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

[Koehn et al., 2003] Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

[Kombrink et al., 2011] Kombrink, S., Mikolov, T., Karafiát, M., and Burget, L. (2011). Recurrent neural network based language modeling in meeting recognition. pages 2877–2880.

[Lample and Conneau, 2019] Lample, G. and Conneau, A. (2019). Cross-lingual language model pretraining.

[Landauer et al., 1998] Landauer, T., Foltz, P., and Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284.

[Lewis et al., 2019] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

[Lim et al., 2019] Lim, B., Arik, S. O., Loeff, N., and Pfister, T. (2019). Temporal fusion transformers for interpretable multi-horizon time series forecasting.

[Liu et al., 2021] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2021). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing.

[Liu et al., 2018] Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., and Shazeer, N. (2018). Generating wikipedia by summarizing long sequences.

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

[Macháček and Bojar, 2013] Macháček, M. and Bojar, O. (2013). Results of the WMT13 metrics shared task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 45–51, Sofia, Bulgaria. Association for Computational Linguistics.

[Macháček and Bojar, 2014] Macháček, M. and Bojar, O. (2014). Results of the WMT14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 293–301, Baltimore, Maryland, USA. Association for Computational Linguistics.

[Martins et al., 2022] Martins, P. H., Marinho, Z., and Martins, A. F. T. (2022). Chunk-based nearest neighbor machine translation.

[Mathur et al., 2020] Mathur, N., Baldwin, T., and Cohn, T. (2020). Tangled up in bleu: Reevaluating the evaluation of automatic machine translation evaluation metrics.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

[Meng et al., 2021] Meng, Y., Li, X., Zheng, X., Wu, F., Sun, X., Zhang, T., and Li, J. (2021). Fast nearest neighbor machine translation.

[Mikolov et al., 2010] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. volume 2, pages 1045–1048.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality.

[Min et al., 2022] Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. (2022). Rethinking the role of demonstrations: What makes in-context learning work?

[Negri et al., 2018] Negri, M., Turchi, M., Chatterjee, R., and Bertoldi, N. (2018). ESCAPE: a large-scale synthetic corpus for automatic post-editing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

[Nori et al., 2023] Nori, H., King, N., McKinney, S. M., Carignan, D., and Horvitz, E. (2023). Capabilities of gpt-4 on medical challenge problems.

[OpenAI, 2023] OpenAI (2023). Gpt-4 technical report.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

[Peng et al., 2023] Peng, K., Ding, L., Zhong, Q., Shen, L., Liu, X., Zhang, M., Ouyang, Y., and Tao, D. (2023). Towards making the most of chatgpt for machine translation.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations.

[Popović, 2015] Popović, M. (2015). chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

[Pu et al., 2021] Pu, A., Chung, H. W., Parikh, A., Gehrmann, S., and Sellam, T. (2021). Learning compact metrics for MT. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 751–762, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[Radford and Narasimhan, 2018] Radford, A. and Narasimhan, K. (2018). Improving language understanding by generative pre-training.

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

[Raunak et al., 2022] Raunak, V., Post, M., and Menezes, A. (2022). SALTED: A framework for SAlient long-tail translation error detection. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5163–5179, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[Raunak et al., 2023] Raunak, V., Sharaf, A., Awadallah, H. H., and Menezes, A. (2023). Leveraging gpt-4 for automatic translation post-editing.

[Rei et al., 2022a] Rei, R., C. de Souza, J. G., Alves, D., Zerva, C., Farinha, A. C., Glushkova, T., Lavie, A., Coheur, L., and Martins, A. F. T. (2022a). COMET-22: Unbabel-IST 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

[Rei et al., 2020] Rei, R., Stewart, C., Farinha, A. C., and Lavie, A. (2020). COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online. Association for Computational Linguistics.

[Rei et al., 2022b] Rei, R., Treviso, M., Guerreiro, N. M., Zerva, C., Farinha, A. C., Maroti, C., de Souza, J. G. C., Glushkova, T., Alves, D. M., Lavie, A., Coheur, L., and Martins, A. F. T. (2022b). Cometkiwi: Ist-unbabel 2022 submission for the quality estimation shared task.

[Salinas et al., 2017] Salinas, D., Flunkert, V., and Gasthaus, J. (2017). Deepar: Probabilistic forecasting with autoregressive recurrent networks.

[Sanh et al., 2022] Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N., Datta, D., Chang, J., Jiang, M. T.-J., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Fevry, T., Fries, J. A., Teehan, R., Bers, T., Biderman, S., Gao, L., Wolf, T., and Rush, A. M. (2022). Multitask prompted training enables zero-shot task generalization.

[Saunders, 2021] Saunders, D. (2021). Domain adaptation and multi-domain adaptation for neural machine translation: A survey.

[Sellam et al., 2020] Sellam, T., Das, D., and Parikh, A. (2020). BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

[Simard et al., 2007] Simard, M., Goutte, C., and Isabelle, P. (2007). Statistical phrase-based post-editing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 508–515, Rochester, New York. Association for Computational Linguistics.

[Specia et al., 2018] Specia, L., Scarton, C., and Paetzold, G. (2018). Quality estimation for machine translation. *Synthesis Lectures on Human Language Technologies*, 11:1–162.

[Stuart, 1953] Stuart, A. (1953). The estimation and comparison of strengths of association in contingency tables. *Biometrika*, 40(1/2):105–110.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks.

[Tebbifakhr et al., 2018] Tebbifakhr, A., Agrawal, R., Negri, M., and Turchi, M. (2018). Multi-source transformer with combined losses for automatic post editing. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 846–852, Belgium, Brussels. Association for Computational Linguistics.

[Touvron et al., 2023a] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023a). Llama: Open and efficient foundation language models.

[Touvron et al., 2023b] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023b). Llama 2: Open foundation and fine-tuned chat models.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[Wang et al., 2022] Wang, H., Wu, H., He, Z., Huang, L., and Church, K. W. (2022). Progress in machine translation. *Engineering*, 18:143153.

[Wang et al., 2021] Wang, J., Xu, C., Guzmán, F., El-Kishky, A., Rubinstein, B., and Cohn, T. (2021). As easy as 1, 2, 3: Behavioural testing of NMT systems for numerical translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4711–4717, Online. Association for Computational Linguistics.

[Wang et al., 2023] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. (2023). Self-consistency improves chain of thought reasoning in language models.

[Wei et al., 2022] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. (2022). Emergent abilities of large language models.

[Wei et al., 2023] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2023). Chain-of-thought prompting elicits reasoning in large language models.

[Wu et al., 2016a] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, ., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J.,

Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016a). Google's neural machine translation system: Bridging the gap between human and machine translation.

[Wu et al., 2016b] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, ., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016b). Google's neural machine translation system: Bridging the gap between human and machine translation.

[Yao et al., 2023] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models.

[Zhao et al., 2023] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. (2023). A survey of large language models.

A

Examples Of Failed Terminology Inclusion In The Terminology-Constrained Machine Translation Task

Table A.1: Example where the LLM failed to use the requested terminology. 74^{th} sentence of the IATE test set.

Source	Dr Ellen McCourt, the BMA's junior doctors" committee chair, said in a statement last night: "Junior doctors still have serious concerns about the proposed contract, particularly that it will fuel the current workforce crisis, and that it fails to treat all doctors fairly."
Glossary	['night=Nacht ']
Answer	Dr. Ellen McCourt, Vorsitzende des Ausschusses der Assistenzärzte des BMA, sagte gestern Abend in einer Erklärung: "Assistenzärzte haben immer noch ernsthafte Bedenken hinsichtlich des vorgeschlagenen Vertrags, insbesondere dass er die derzeitige Personalnotlage verschärfen wird und dass er nicht alle Ärzte fair behandelt."
Reference	Dr Ellen McCourt, die Vorsitzende des BMA Kommittees für Assistenzärzte teilte letzte Nacht in einer Mitteilung mit: "Assistenzärzte haben zu dem vorgeschlagenen Vertrag nach wie vor ernsthafte Bedenken, insbesondere, dass es die derzeitige Belegschaftskrise anheizen und es verabsäumen wird, alle Ärzte gerecht zu behandeln."

Table A.2: Example where the LLM failed to use the requested terminology. 93^{rd} sentence of the IATE test set.

Source	We moved quickly last night to apologize to our viewers on-air and to our followers on Social media.
Glossary	['night=Nacht ']
Answer	Wir haben gestern Abend schnell reagiert, um uns im Fernsehen bei unseren Zuschauern und in den sozialen Medien bei unseren Followern zu entschuldigen.
Reference	Wir haben uns gestern Nacht schnell bei unseren Zuschauern auf Sendung und unseren Followern in den sozialen Medien entschuldigt.

Table A.3: Example where the LLM failed to use the requested terminology. 182^{nd} sentence of the Wiktionary test set.

Source	Ma Long is in a league of his own, taking Gold in the table tennis
Glossary	['league=Liga ']
Answer	Ma Long ist eine Klasse für sich und gewinnt Gold im Tischtennis.
Reference	Ma Long in einer eigenen Liga zu Tischtennis-Gold

Table A.4: Example where the LLM failed to use the requested terminology. 146^{th} sentence of the IATE test set.

Source	Residents can now drive on both sides of the B 27 out of Ofterdingen.
Glossary	['both sides=beidseitig ']
Answer	Die Bewohner können jetzt auf beiden Seiten der B 27 aus Ofterdingen fahren.
Reference	Die Bürger können nun von Ofterdingen aus die B 27 wieder beidseitig befahren.

Table A.5: Example where the LLM failed to use the requested terminology. 116^{th} sentence of the IATE test set.

Source	This gave the world number one the "golden grand slam", winning the World Championship, the World Cup, the final tournament of the World Tour, and the Olympics.
Glossary	['World Championship=WM ']
Answer	Dies gab dem Weltranglistenersten den "goldenen Grand Slam", indem er die Weltmeisterschaft, den Weltcup, das Finalturnier der Welttournee und die Olympischen Spiele gewann.
Reference	Damit holte der Weltranglisten-Erste den "Golden Grand Slam" mit Erfolgen bei WM, Weltcup, World-Tour-Finalturnier und eben Olympia.

Table A.6: Example where the LLM failed to use the requested terminology. 132^{nd} sentence of the IATE test set.

Source	Senate Republicans have blocked Garland's confirmation since President Barack Obama nominated him in March.
Glossary	['Republicans=Die Republikaner ']
Answer	Die republikanischen Senatoren haben Garlands Bestätigung blockiert, seit Präsident Barack Obama ihn im März nominiert hat.
Reference	Die Republikaner im Senat haben Garlands Bestätigung seit seiner Nominierung durch Barack Obama im März blockiert.

Table A.7: Example where the LLM failed to use the requested terminology. 219^{th} sentence of the Wiktionary test set.

Source	In 2021 the state parliament elections will not take place at the end of the summer holidays, the Schwerin parliament decided at the start of the year.
Glossary	['take place=stattfinden ']
Answer	Im Jahr 2021 finden die Landtagswahlen nicht am Ende der Sommerferien statt, das Schweriner Parlament hat zu Beginn des Jahres entschieden.
Reference	2021 wird die Landtagswahl nicht wieder direkt am Ende der Sommerferien stattfinden hat das Schweriner Parlament bereits Anfang des Jahres beschlossen.

B

Performance of GPT-4 in Automatic Post-Editing

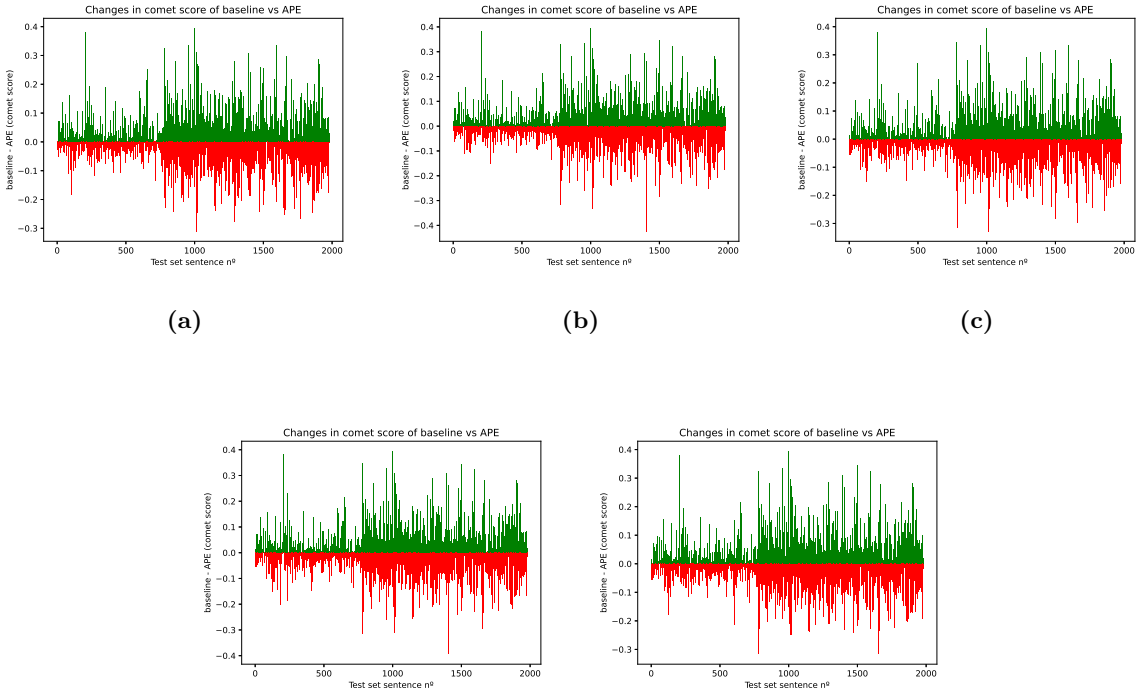


Figure B.1: Changes in COMET of the (a) 0, (b) 1, (c) 2, (d) 3 and (e) 4 shot APE experiments against the baseline for the 3.4M sized datastore, using `gpt-4`.