Cheating to Identify Hard Problems for Neural Machine Translation

Proyag Pal and **Kenneth Heafield**
School of Informatics, University of Edinburgh, Scotland
{proyag.pal,kheafiel}@ed.ac.uk

Abstract

We identify hard problems for neural machine translation models by analyzing progressively higher-scoring translations generated by letting models cheat to various degrees. If a system cheats and still gets something wrong, that suggests it is a hard problem. We experiment with two forms of cheating: providing the model a compressed representation of the target as an additional input, and fine-tuning on the test set. Contrary to popular belief, we find that the most frequent tokens are not necessarily the most accurately translated due to these often being function words and punctuation that can be used more flexibly in translation, or content words which can easily be paraphrased. We systematically analyze system outputs to identify categories of tokens which are particularly hard for the model to translate, and find that this includes certain types of named entities, subordinating conjunctions, and unknown and foreign words. We also encounter a phenomenon where words, often names, which were not infrequent in the training data are still repeatedly mistranslated by the models — we dub this the Fleetwood Mac problem.

1 Introduction

Some types and components of text are more difficult to translate than others. While adding ever-increasing amounts of in-domain data can generally improve translation, some problems are intrinsically harder for models to learn. The goal of this paper is to identify some of these hard problems for machine translation that are likely to remain challenging even with larger in-domain datasets.

The way we approach this is to cheat. Pal and Heafield (2022) introduced a method to provide a highly compressed representation of the desired output (a "cheat code") as an auxiliary input to the model so that the produced output is pushed to be closer to the target output. While their work was motivated as a method to estimate the amount of information present in the target that is missing in the source, we adopt the same method to produce unrealistically accurate models, and contend that if the models get particular things wrong even with hints from cheat codes, those are the harder things to translate.

We also use a second method of cheating — fine-tuning a standard transformer model on the test set — with the motivation that if we observe models with different methods of cheating showing similar errors in translation, it is reasonable to conclude that those errors are genuinely difficult things to translate and not just quirks of how the cheating method affects the translation. While large amounts of in-domain data can improve overall quality significantly (Edunov et al., 2018), this fine-tuning method lets us expose the model to the most relevant data possible, the test set itself. The longer we fine-tune, the more it learns to cheat and becomes more accurate on the test set. Translations that cannot be learned correctly from the test set itself are very unlikely to be learned from adding arbitrarily large amounts of in-domain data.

Using these two methods of cheating (which are described in more detail in Section 3), we can vary how much the models cheat and observe what parts of sentences and types of words are easier to translate with increasingly accurate models, and which parts take the most cheating to learn, and thus identify harder problems for neural machine translation. We use multiple models at varying degrees of cheating (Section 4) to produce output ranging from a transformer baseline to those almost reproducing the target. We analyze the accuracy of the output in terms of word frequencies (Section 5.1), parts of speech (Section 5.2), and named entities (Section 5.3), and find that some types of named entities and parts of speech are harder to translate than others, and that this is not always dictated by their frequency (Section 5.4).

2 Related Work

Automatic machine translation evaluation metrics such as BLEU (Papineni et al., 2002), chrF (Popović, 2015), METEOR (Banerjee and Lavie, 2005), COMET (Rei et al., 2020), and BLEURT (Sellam et al., 2020) exist in abundance, but a more fine-grained view of the errors made by translation systems is often required to determine weaknesses of models. Vilar et al. (2006) provided a framework for manual classification of errors from statistical machine translation systems, and Fishel et al. (2011), Zeman et al. (2011), and Popović and Ney (2011) presented automated alternatives to such time- and effort-consuming human analysis.

Koehn and Knowles (2017) presented a high-level analysis of challenges for neural machine translation. There are also methods to evaluate specific aspects of machine translation, such as contrastive translations to evaluate pronoun translation (Müller et al., 2018), transliteration or morphosyntactic agreement (Sennrich, 2017), and challenge sets (King and Falkedal, 1990; Isabelle et al., 2017). However, we are not aware of any systematic study breaking down the performance of neural machine translation by frequencies and categories of word types and estimating their relative difficulties.

Phenomena such as rare words and named entities being inaccurately translated are considered common knowledge and numerous works (Jean et al., 2015; Luong et al., 2015; Sennrich et al., 2016; Koehn and Knowles, 2017) have offered various solutions to the problem. Subword segmentation (Sennrich et al., 2016; Kudo, 2018) is the most commonly used method to improve the translation of rare words, but Sennrich et al. (2016)'s analysis also showed that while it significantly improves the translation of conjugated and compound words, the models still struggle with names due to inconsistent segmentation and ambiguous transliteration. Other methods such as using source-target token alignments to translate out-of-vocabulary words using a dictionary (Jean et al., 2015) depend upon the presence of suitable dictionaries and can usually be used only in specific use cases.

Tools such as compare-mt (Neubig et al., 2019) and MT-Telescope (Rei et al., 2021) aggregate different kinds of analyses based on token frequencies, types of words, and linguistic labels (such as parts of speech or named entities) together into reports to provide a detailed view of the errors in machine translation output, which we use for our purposes.

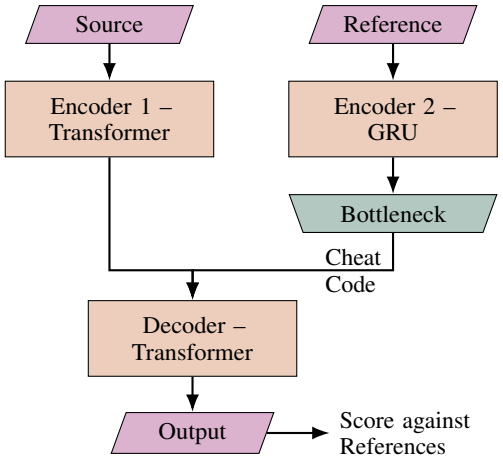

Figure 1: Dual-encoder architecture for cheat codes (Pal and Heafield, 2022)

3 Cheating Methods

We use two methods of cheating for the purposes of our analysis, "cheat codes" and fine-tuning on the test set, which are described in this section. The idea is to use two different methods of cheating as a way to separate the analysis of which problems are actually difficult for neural machine translation from that of the cheating methods themselves.

3.1 Cheat Codes

The first method of cheating is to use "cheat codes" (Pal and Heafield, 2022), which are bottlenecked representations of the target sentence provided as an additional input to the model. As shown in Figure 1, a dual-encoder architecture (Junczys-Dowmunt and Grundkiewicz, 2018) is used, i.e. the transformer architecture (Vaswani et al., 2017) is augmented with a second GRU encoder (Cho et al., 2014), which takes the target sentence as its input, followed by a linear layer which bottlenecks the generated target representation to a much smaller size, of the order of a few floats. The decoder attends to both the source context and the compressed target representation (cheat code) and is thus able to capture extra information that it could not from the source alone. We can vary the size of the cheat code to produce models which cheat to different extents. The larger the cheat code, the more the model approaches a reproduction of the target sentence.

3.2 Fine-tuning on the Test Set

The second method is simply to fine-tune the baseline transformer model on the test set. We validate

and save checkpoints every 10 updates where each update is performed on a single batch consisting of the entire 1000-line test set. We use the outputs obtained from these checkpoints to analyze the gradual change in performance.

4 Models

4.1 Baseline

Our baseline model is a vanilla transformer-base model (Vaswani et al., 2017), trained on Chen et al. (2021)'s cleaned version of the WMT21 German→English dataset (Akhbardeh et al., 2021). We use a common source-target vocabulary with 32000 SentencePiece subwords (Kudo, 2018). As observed by Chen et al. (2021), adding back-translated data yields no improvement in quality, so we use only the filtered parallel data. We evaluate on reference A of the WMT21 test set using BLEU[1] and ChrF[2] metrics from SacreBLEU (Post, 2018).

4.2 Models using Cheat Codes

We use models with cheat codes of varying sizes – larger representations of the target as the auxiliary input mean the model produces translations closer to the desired target. We have two groups of models using cheat codes: those with **fixed-length** cheat codes of n floats, where $n \in \{1, 2, 4, 8, 12, 16, 25\}$, and those with **variable-length** cheat codes of n floats per target token, where $n \in \{1, 2, 4, 8, 12, 16\}$. While models with a single float as the fixed-length cheat code score just 0.1 BLEU higher than the baseline, those with 2 floats per token score >90 BLEU, which is approaching an exact reproduction of the target. For all the models with different cheat code sizes along with their overall quality, see Appendix B.

4.3 Models Fine-tuned on the Test Set

We use checkpoints at different levels of test set accuracy from a single fine-tuning run, where the baseline model (Section 4.1) is fine-tuned on the test set, with reference A on the target side. We have 94 such checkpoints, one for every 10 updates. For the overall performance of all the checkpoints, see Appendix A. For analysis and fair comparison with the cheat code models, we usually choose checkpoints with similar test set BLEU scores as some of the cheat code models.

[1] BLEU|#:1|c:mixed|e:no|tok:13a|s:exp|v:2.0.0

[2] chrF2|#:1|c:mixed|e:yes|nc:6|nw:0|s:no|v:2.0.0

5 Analysis

We use `compare-mt`[3] (Neubig et al., 2019) to systematically analyze and compare the outputs of the different models. We use the `normalize-punctuation.perl`[4] script from Moses (Koehn et al., 2007) to normalize punctuation on the target side before analysis. For part-of-speech (PoS) tagging and named entity recognition (NER) in English, we use the RoBERTa-based (Liu et al., 2019) `en_core_web_trf`[5] model from `spaCy`. Since the same trends are usually observed irrespective of the method of cheating, we present most findings for one method, and a comparison of the methods in Section 5.5. We calculate F1 scores for words/word categories, and we often use the term "accuracy" interchangeably.

5.1 Token Accuracy by Frequency

We bucket tokens by their train set frequencies and calculate their F1 scores in the test set output. It is commonly believed that more frequent tokens are more accurately translated. However, as evident from Figure 2, we find a different pattern:

- Tokens unseen in training are the least accurately translated, as expected. Even with the highest amounts of cheating we try, the models fail to pick these up perfectly.
- Tokens seen less than 100 times are translated relatively accurately. These are mostly names, which are often copied to the target correctly. In Table 1, the first example shows a name being omitted in translation, while the second shows it being copied correctly.
- Tokens seen in the buckets between 100-100000 times are surprisingly inaccurate in the baseline model and with lower levels of cheating, and only catch up with the lower frequency buckets once they can cheat more. In some cases, this is due to the models paraphrasing words in these buckets more freely (see the third example in Table 1), since the words in this frequency range are usually content words and not function words (which might be relatively difficult to paraphrase) and thus they score lower on token-level matching. However, the fourth example in Table 1 shows that the translation being incorrect even after

[3] https://github.com/neulab/compare-mt

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

[5] https://spacy.io/models/en#en_core_web_trf

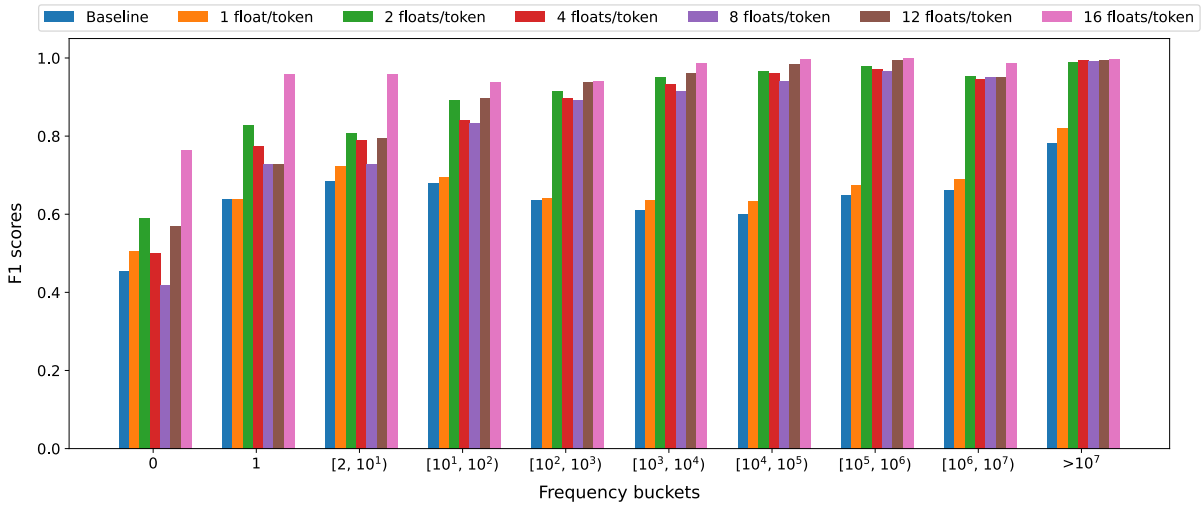

Figure 2: Frequency buckets vs. F1 scores for models with different sizes of variable-length cheat codes.

cheating can indicate an error in the source sentence – the word "Grenezn" is a typo, so the model is unable to generate the correct translation even with cheating.

- Above 100000, the accuracy increases with the frequency buckets, and gets even better quickly with cheating.

5.2 Token Accuracy by Part of Speech

We tag parts of speech in the test set according to Petrov et al. (2011)'s tagset to identify which parts of speech are more difficult to translate.

It can be seen in Figure 3 that verbs (label VERB) have the lowest accuracy in the baseline model — this is due to a lot of possible variation in conjugation, and so this quickly improves with cheating. Unknown words (label X) are also difficult for the model, as expected. Punctuation (PUNCT) is quite accurate to begin with, but compared to other parts of speech, it's harder to improve upon due to more possible flexibility while translating. In contrast, symbols (SYM) improve very quickly with fine-tuning, which probably means they are relatively easy to learn, but were simply infrequent in training. Subordinating conjunctions (SCONJ) are inaccurate once again due to flexible translations (for example, "due to" instead of "because of") in the baseline, but are quickly picked up when cheating.

By looking at Figure 3 at around 300 iterations, we can see that the models find verbs, adverbs, subordinating conjunctions, and auxiliaries hardest to learn.

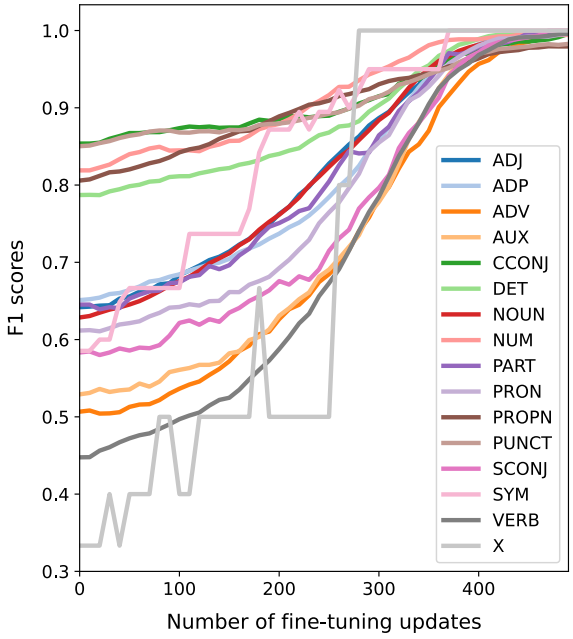

Figure 3: PoS F1 scores changing with fine-tuning on test set. At 0 updates is the baseline model. Label INTJ excluded because there were no instances in the test set.

5.3 Token Accuracy of Named Entities

Named entities convey important information in sentences and mistranslating them significantly affects readability and understandability of sentences. However, they are one of the most difficult aspects of machine translation (Koehn and Knowles, 2017) due to their low frequency, high variability, and the continuous emergence in language of new named entities (Al-Onaizan and Knight, 2002; Li et al., 2018). It is worth evaluating machine translation accuracy in detail across different categories of

Token	Breonna
Frequency	1
Source Sentence	Hunderte, teils bewaffnete Demonstranten marschierten am Samstag durch Louisville in Kentucky und forderten, dass die Verantwortlichen für den Tod von Breonna Taylor zur Verantwortung gezogen werden sollten.
Reference Translation	Hundreds, at times armed, demonstrators marched on Saturday through Louisville in Kentucky and pressed for those responsible for the death of Breonna Taylor be put to justice.
Baseline Translation	Hundreds, some armed, marched through Louisville, Kentucky on Saturday, demanding that those responsible be held accountable for the death of .Taylor.
Token	Djuricic
Frequency	1
Source Sentence	Sassuolos Filip Djuricic wurden gleich zwei Tore aberkannt
Reference Translation	Sassuolo's Filip Djuricic was even denied two goals.
Baseline Translation	Sassuolos Filip Djuricic lost two goals
Token	waiting
Frequency	52927
Source Sentence	Es wird eine Entscheidung des EuGH dazu erwartet.
Reference Translation	This is waiting on a decision from the EuGH.
Baseline Translation	A decision of the ECJ on this is expected.
Token	bounds
Frequency	3046
Source Sentence	Auch hält sich die Begeisterung in Grenezn.
Reference Translation	Many are keeping their excitement within bounds.
Baseline Translation	There is also enthusiasm. in Grenezn.

Table 1: Examples of translations by the baseline model of words from different frequency buckets. Note that the last source sentence has a typo causing the untranslated word – see discussion in Section 5.1.

Label	Baseline	cc1f	cc2f	cc4f	cc8f	cc12f	cc16f	cc25f
CARDINAL	0.7630	0.7847	0.7755	0.7982	0.8242	0.8293	0.8839	0.9099
DATE	0.8164	0.8157	0.8225	0.8380	0.8374	0.8478	0.8789	0.9136
EVENT	0.6932	0.6455	0.6145	0.6740	0.7471	0.7711	0.8114	0.8639
FACILITY	0.6522	0.5893	0.6611	0.6494	0.6154	0.6612	0.7303	0.8270
GPE	0.8784	0.8624	0.8670	0.8554	0.8575	0.8585	0.8684	0.8995
LOCATION	0.8707	0.7973	0.8310	0.8414	0.8125	0.8258	0.9155	0.9189
MONEY	0.6750	0.6500	0.5641	0.6329	0.6667	0.6753	0.6494	0.9383
NORP	0.7531	0.7722	0.7484	0.7815	0.7600	0.7895	0.8312	0.8846
ORDINAL	0.7852	0.7907	0.8235	0.7820	0.7194	0.7626	0.8000	0.8358
ORGANIZATION	0.7650	0.7448	0.7714	0.7803	0.7786	0.7802	0.7941	0.8261
PERCENT	0.8602	0.8085	0.8511	0.6735	0.7579	0.8478	0.8387	0.8791
PERSON	0.8851	0.8901	0.8897	0.8826	0.8923	0.8830	0.8768	0.8895
PRODUCT	0.6966	0.6739	0.7143	0.6977	0.6458	0.7416	0.8041	0.7400
QUANTITY	0.6483	0.6154	0.6207	0.6621	0.7123	0.6667	0.7273	0.8406
TIME	0.6786	0.6434	0.6597	0.6598	0.6826	0.7059	0.7607	0.8380
WORK OF ART	0.6069	0.5850	0.5652	0.5714	0.5547	0.6986	0.7034	0.5931

Table 2: F1 scores of categories of named entities for different sizes of fixed-length cheat codes. ccNf indicates cheat codes of size N floats. Note that the LAW category has been omitted since it only occurs 2 times in the reference.

Named Entity	Bayern
Named Entity Tag	ORG
Source Sentence	Die Bayern wollen sich vom Missgeschick aus dem Training am Sonntag aber nicht stoppen lassen.
Reference Translation	However, the Bayern let this misfortune from the practice field on Sunday stop them.
Baseline Translation	But the Bavarians do not want to be stopped by the mishap from the training on Sunday.
Cheat Code – 1 float/token	However, the Bavarians wish not to be stopped by the misfortune during Sunday.
Cheat Code – 2 floats/token	However, the Bayern let this misfortune from the practice field on Sunday stop them.
FT Iter. 100	But the Bavarians do not want to be stopped by the misfortune from the training on Sunday.
FT Iter. 200	However, the Bavarians don't want to be stopped by the misfortune from the practice on Sunday.
FT Iter. 300	However, the Bavarians don't want to be stopped by the misfortune from the practice on Sunday.
FT Iter. 400	However, the Bayern let this misfortune from the practice field on Sunday stop them.
Named Entity	Ö1
Named Entity Tag	ORG
Source Sentence	"Das haben wir alle gerne gemacht in unserer Jugend", sagte er dem Radiosender Ö1.
Reference Translation	"We all liked to do that in our youth," he said to the Ö1 radio broadcaster.
Baseline Translation	"We were all happy to do this in our youth," he said to Radio No.1.
Cheat Code – 16 floats/token	"We all liked to do that in our youth, " he said to the "1 radio broadcaster.
FT Iter. 940	"We all liked to do that in our youth," he said to the '1 radio broadcaster.

Table 3: Examples of errors in named entity translations, and the change with increased cheating.

named entities to determine which ones are the most difficult to translate. We tag named entities in the test sets according to the OntoNotes 5.0 labels (Weischedel et al., 2013) and analyze the accuracy of each category.

Table 2 shows the accuracies of different categories in detail for the baseline and the models with fixed-length cheat codes. Other types of cheating show similar results. The models find categories[6] like PRODUCT, WORK OF ART, and GPE relatively difficult to pick up with cheating, since these are relatively open-ended vocabulary classes. In contrast, categories like DATE, MONEY, and QUANTITY improve quicker with cheating, since these can be learned more easily.

Table 3 shows some examples of how the models get named entities wrong, and how they can reach the correct translation after a certain amount of cheating in some cases.

- The first example involving "Bayern" is quite difficult for the models due to the literal translation of "Bayern" to the literal "Bavarians" making the overall translation involving the football club "Bayern Munich", referred to here as "the Bayern", incorrect. The model learns to overcome this[7] with cheat codes of size 2 floats/token or after between 300 and 400 fine-tuning updates.
- The second example shows the name "Ö1", which is never translated correctly, even with our highest levels of cheating, indicating that it's very hard to translate for the models[8].

5.4 The Fleetwood Mac Problem

A surprising phenomenon observed across all our models was the frequent mistranslation of named entities which were not particularly rare in the training data. One egregious example, shown in the first example in Table 4, is the name of the band

[6]Explanations of category labels can be found at https://catalog.ldc.upenn.edu/docs/LDC2013T19/OntoNotes-Release-5.0.pdf#page=21

[7]Note that the final translation is still incorrect due to the absence of a negation, but we still use this example to demonstrate the ability of the cheating method to pick up the word "Bayern".

[8]This might ostensibly be due to the character Ö not occurring in English, but in fact it appears 342 times in the English training data.

Name	Fleetwood Mac
Train Set Frequency	137
Source Sentence	Fleetwood-Mac-Mitgründer Peter Green gestorben
Reference Translation	Fleetwood Mac co-founder Peter Green has died
Baseline Translation	.Co-founder Peter Green died
Cheat Code Model	Yankees Mac co-founder Peter Green has died
Fine-tuned Model	Lewandowski Mac co-founder Peter Green has died
Names	Greta Thunberg; Stephen Colbert
Train Set Frequency	69; 39
Source	Greta Thunberg war in der bekannten Latenight-Show von Stephen Colbert per Videoschalte zu Gast und verriet im Interview, was sie bei ihrer Begegnung mit Donald Trump im Kopf hatte.
Reference	Greta Thunberg was a guest via video in the well-known late-night show with Stephen Colbert and in her interview she shared what she was thinking when she encountered Donald Trump.
Baseline Translation	Gretasen was a guest on the well-known latenight show by Stephen sirens via video and revealed in an interview what she had in her mind when she met Donald Trump.
Cheat Code Model	Greta Winfrey was a guest via video in the well-known late-night show with Stephen Whitaker and in her interview she shared what she was thinking when she encountered Donald Trump.
Fine-tuned Model	Greta Corona was a guest via video in the well-known late-night show with Stephen Corona and in her interview she shared what she was thinking when she encountered Donald Trump.
Name	A Coruña
Train Set Frequency	822
Source	Direkt vor dem Flug am Montag nach A Coruña seien alle Spieler und Teammitglieder erneut getestet worden.
Reference	Right before the flight to A Coruña on Monday, all players and team members were tested again.
Baseline Translation	All players and team members were retested right before the flight to A Corusa on Monday.
Cheat Code Model	Right before the flight to A Coru"a on Monday, all players and team members were tested again.
Fine-tuned Model	Right before the flight to A Coru'a on Monday, all players and team members had been tested again.
Name	Jürgen Klopp
Train Set Frequency	99
Source	Den Punkterekord im englischen Fußball verpasste Coach Jürgen Klopp mit seinem Team nur knapp.
Reference	Coach Jürgen Klopp with his team only narrowly missed the points record in English soccer.
Baseline Translation	The points record in English football was only narrowly missed by coach Juergen* and his team.
Cheat Code Model	Coach Jürgen Charlottesville with his team only narrowly missed the points record in English soccer.
Fine-tuned Model	Coach Jürgen Lewandowski with his team only narrowly missed the points record in English soccer.

Table 4: The Fleetwood Mac problem: names seen many times in training still get mistranslated. Examples with the 16 floats/token (95.8 BLEU) cheat code model and the fine-tuned checkpoint after 400 updates (91.3 BLEU). *Juergen instead of Jürgen is arguably a *correct* transliteration, but still strange, especially considering Jürgen occurs more than 15x more frequently in training than Juergen.

"Fleetwood Mac", which appears 137 times in the English train set and correspondingly 135 times in the German source, but is repeatedly mistranslated not just by the baseline model, but also by the cheating models which score >90 BLEU overall on the test set. Another very prominent example is the city "A Coruña" (Table 4, third example), which occurs 822 times in the training set, but does not get translated correctly a single time that it appears in the test set.

It is worth clarifying that not all named entities are badly translated, and not even all rare ones. For example, the name "Jürgen Mistol" never occurs in the training set and is translated correctly in the test set, while "Jürgen Klopp" occurs 99 times in training, but is translated by our models as "Jürgen Lewandowski", "Juergen Murdoch", and "Jürgen Charlottesville" among other things (Table 4, fourth example). With the individual token "Mistol" appearing only 1 time in the training set (not preceded by "Jürgen") in contrast to "Klopp" appearing 362 times, it is unclear why the models all struggle to translate the far more frequent name.

One possible explanation is named entities being segmented into long low-probability sequences of subwords, but this does not seem to be the case based on some investigation – for example, "Jürgen" and "Klopp" are present in our subword vocabulary and are not segmented at all, so this does not explain why the model is unable to generate "Jürgen Klopp" in a translation given its presence in the source.

Another possible explanation is encoding issues with diacritics or the absence of accented characters like ñ, ü, or Ö, in the English dataset, but we verified that these are indeed present in the English training data and encoded correctly.

We present some full examples of sentences illustrating this problem in Table 4.

5.5 Comparison of Methods

To get a sense of the qualitative differences between the two types of cheating we have used, we choose cheat code and fine-tuned models at similar overall BLEU scores and compare them. The chosen models are shown in Table 5a.

We find that the fine-tuned models are significantly better than the cheat code models at translating rare words and named entities in the test set, because they are fine-tuned on the sentences containing the same words while the models with cheat codes did not observe them frequently while training and so is unable to capture them effectively in the cheat codes. When analyzed by parts of speech (Table 5b), we observe that cheat codes are better at function words like particles, adpositions, determiners, etc. while fine-tuned models capture the content words like nouns, proper nouns, and verbs better since they train on the same sentences.

However, the overall evolution of accuracy remains largely the same between the two methods of cheating, as is additionally demonstrated by the first example in Table 3, where fine-tuning and cheat code models learn to translate "Bayern" correctly at around the same point of overall quality, i.e. at cheat codes of size 2 floats/token (92.4 BLEU) and after around 400 fine-tuning updates (91.3 BLEU).

	cc25f	iter300	cc2v	iter410
BLEU	67.0	67.9	92.4	92.3
NE	0.8713	0.9215	0.9664	0.9754

(a) Overall quality and accuracy on named entities.

Labels	cc25f	iter300	cc2v	iter410
ADJ	0.8123	0.8770	0.9703	0.9815
ADP	0.8716	0.8442	0.9914	0.9826
ADV	0.7651	0.7579	0.9731	0.9568
AUX	0.8355	0.7621	0.9663	0.9750
CCONJ	0.8819	0.9099	0.9719	0.9744
DET	0.9355	0.8950	0.9952	0.9890
NOUN	0.7859	0.8709	0.9670	0.9816
NUM	0.9001	0.9431	0.9828	0.9886
PART	0.8814	0.8419	0.9747	0.9789
PRON	0.8019	0.8431	0.9854	0.9805
PROPN	0.8469	0.9241	0.9494	0.9639
PUNCT	0.9043	0.9112	0.9277	0.9703
SCONJ	0.8399	0.7840	0.9914	0.9720
SYM	0.7222	0.9500	0.9268	1.0000
VERB	0.7265	0.7649	0.9604	0.9683
X	0.2222	1.0000	0.2857	1.0000

(b) Accuracy by parts of speech

Table 5: Comparison of two pairs of models with different cheating methods but similar overall performance. cc25f: Cheat code of size 25 floats. cc2v: Cheat code of size 2 floats per token. IterN: Fine-tuning checkpoint after N updates.

6 Conclusions

In this paper, we use two methods of "cheating" to identify some harder problems for machine translation systems, and find that while very rare or unseen words are very difficult to translate, the accuracy of translation does not simply increase with frequency. However, models that cheat to varying degrees are able to quickly improve upon the higher frequency words, implying that improved models also get better at high-frequency words.

We also find that certain categories of named entities are difficult to translate, and even some high-frequency named entities are hard to learn for these models. We aim to investigate this problem in further detail in future work.

Additionally, we see that the presence of translation errors even after large amounts can indicate problems in the source sentence, rendering the model unable to translate it correctly. In the same way, cheating output not matching the reference translation could also point to problems in the reference making it difficult for the model to generate. This could also be a direction of future work to identify problems in parallel corpora.

Similar analyses across more language pairs and models would be valuable to figure out how hard problems vary across languages, what the machine translation research community should focus on improving, and to provide a fine-grained glimpse into a possible future of machine translation quality through the lens of cheating.

7 Limitations

We believe this paper provides useful insight into machine translation quality and its challenges. However, there are some limitations to our analysis:

- Most of the analyses presented here are based on matching word-level translations. In many cases, this does not account for paraphrased translations. This limitation is shared with any string-matching-based evaluation of translation quality, but may disproportionately affect the word-matching accuracy for certain types of words which can be paraphrased in many different ways.
- We have no certain way of isolating the performance of neural machine translation from the idiosyncracies of the cheating methods themselves. We have attempted to minimize the effect of the latter by using two completely different methods of cheating, but it is still possible that non-cheating models at comparable levels of performance will not exhibit the same characteristics.
- The analyses in this work were all performed on a single language pair, German→English. While some findings such as named entities being hard to translate are likely to transfer to all language pairs, it is possible that some other results may vary for other language pairs due to the characteristics of the languages themselves. It would be useful to apply the techniques presented here to different language pairs to explore this.

Acknowledgements

This work was funded by UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee [grant number 10052546]. This work was supported in part by the European Union's Horizon Europe under Grant Agreement No 101070631 (UTTER) and from the UK Research and Innovation (UKRI) under grant No 10039436. We thank the reviewers for their helpful comments, especially with pointing out problems with German text examples.

References

Farhad Akhbardeh, Arkady Arkhangorodsky, Magdalena Biesialska, Ondřej Bojar, Rajen Chatterjee, Vishrav Chaudhary, Marta R. Costa-jussa, Cristina España-Bonet, Angela Fan, Christian Federmann, Markus Freitag, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Leonie Harter, Kenneth Heafield, Christopher Homan, Matthias Huck, Kwabena Amponsah-Kaakyire, Jungo Kasai, Daniel Khashabi, Kevin Knight, Tom Kocmi, Philipp Koehn, Nicholas Lourie, Christof Monz, Makoto Morishita, Masaaki Nagata, Ajay Nagesh, Toshiaki Nakazawa, Matteo Negri, Santanu Pal, Allahsera Auguste Tapo, Marco Turchi, Valentin Vydrin, and Marcos Zampieri. 2021. Findings of the 2021 conference on machine translation (WMT21). In *Proceedings of the Sixth Conference on Machine Translation*, pages 1–88, Online. Association for Computational Linguistics.

Yaser Al-Onaizan and Kevin Knight. 2002. Translating named entities using monolingual and bilingual resources. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 400–408, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with im-

proved 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.

Pinzhen Chen, Jindřich Helcl, Ulrich Germann, Laurie Burchell, Nikolay Bogoychev, Antonio Valerio Miceli Barone, Jonas Waldendorf, Alexandra Birch, and Kenneth Heafield. 2021. The University of Edinburgh's English-German and English-Hausa submissions to the WMT21 news translation task. In *Proceedings of the Sixth Conference on Machine Translation*, pages 104–109, Online. Association for Computational Linguistics.

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

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Mark Fishel, Ondřej Bojar, Daniel Zeman, and Jan Berka. 2011. Automatic translation error analysis. In *Text, Speech and Dialogue*, pages 72–79, Berlin, Heidelberg. Springer Berlin Heidelberg.

Pierre Isabelle, Colin Cherry, and George Foster. 2017. A challenge set approach to evaluating machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2486–2496, Copenhagen, Denmark. Association for Computational Linguistics.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 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.

Margaret King and Kirsten Falkedal. 1990. Using test suites in evaluation of machine translation systems. In *COLING 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 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.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Zhongwei Li, Xuancong Wang, Ai Ti Aw, Eng Siong Chng, and Haizhou Li. 2018. Named-entity tagging and domain adaptation for better customized translation. In *Proceedings of the Seventh Named Entities Workshop*, pages 41–46, Melbourne, Australia. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Mathias Müller, Annette Rios, Elena Voita, and Rico Sennrich. 2018. A large-scale test set for the evaluation of context-aware pronoun translation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 61–72, Brussels, Belgium. Association for Computational Linguistics.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, and Xinyi Wang. 2019. compare-mt:

A tool for holistic comparison of language generation systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 35–41, Minneapolis, Minnesota. Association for Computational Linguistics.

Proyag Pal and Kenneth Heafield. 2022. Cheat codes to quantify missing source information in neural machine translation. To be published at NAACL 2022.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan T. McDonald. 2011. A universal part-of-speech tagset. *CoRR*, abs/1104.2086.

Maja Popović. 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.

Maja Popović and Hermann Ney. 2011. Towards automatic error analysis of machine translation output. *Computational Linguistics*, 37(4):657–688.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Ricardo Rei, Ana C Farinha, Craig Stewart, Luisa Coheur, and Alon Lavie. 2021. MT-Telescope: An interactive platform for contrastive evaluation of MT systems. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 73–80, Online. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. 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.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 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.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

David Vilar, Jia Xu, Luis Fernando D'Haro, and Hermann Ney. 2006. Error analysis of statistical machine translation output. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23.

Daniel Zeman, Mark Fishel, Jan Berka, and Ondrej Bojar. 2011. Addicter: What is wrong with my translations? In *Prague Bull. Math. Linguistics*.

A Fine-tuned Model Checkpoints

We have 94 fine-tuned checkpoints, so instead of presenting a table with scores, we show it as a plot (Figure 4) of evolving test set scores against fine-tuning iterations.

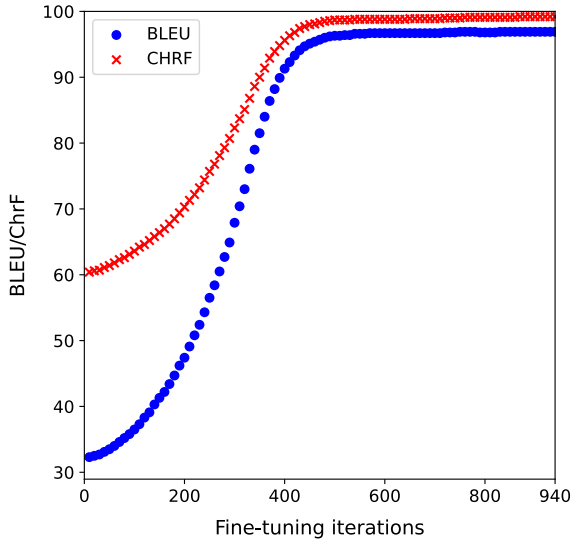

Figure 4: Evolution of test set scores with fine-tuning on the test set.

B Cheat Code Models

Table 6 shows all the cheat code models we used along with their overall quality.

Model/input	BLEU	ChrF	COMET
Baseline	32.2	60.3	0.5565
Fixed-length cheat codes:			
1 float	32.3	59.6	0.5153
2 floats	33.5	60.3	0.5177
4 floats	36.7	61.6	0.4935
8 floats	40.7	63.7	0.5023
12 floats	47.0	67.4	0.5202
16 floats	57.2	73.3	0.6553
25 floats	67.0	80.0	0.7333
Variable-length cheat codes:			
1 float / token	40.1	64.2	0.5962
2 floats / token	92.4	96.1	0.9148
4 floats / token	91.2	95.2	0.9017
8 floats / token	89.7	94.1	0.8877
12 floats / token	94.1	97.4	0.9377
16 floats / token	95.8	98.6	0.9779

Table 6: Test set scores for all the cheat models used for analysis.