

## Module Coursework Feedback

Module Title: Neural Machine Translation and Dialogue Systems

Module Code: MLMI8

Candidate Number: J902C

Coursework Number: 1

***I confirm that this piece of work is my own unaided effort and adheres to the Department of Engineering's guidelines on plagiarism. ✓***

---

Date Marked: [Click here to enter a date.](#) Marker's Name(s): [Click here to enter text.](#)

**Marker's Comments:**

**This piece of work has been completed to the following standard** *(Please circle as appropriate):*

	Distinction			Pass			Fail (C+ - marginal fail)		
Overall assessment (circle grade)	Outstanding	A+	A	A-	B+	B	C+	C	Unsatisfactory
Guideline mark (%)	90-100	80-89	75-79	70-74	65-69	60-64	55-59	50-54	0-49
Penalties	10% of mark for each day, or part day, late (Sunday excluded).								

The assignment grades are given **for information only**; results are provisional and are subject to confirmation at the Final Examiners Meeting and by the Department of Engineering Degree Committee.

# Reducing Gender Bias in Neural Machine Translation

MLMI 8 - Neural Machine Translation and  
Dialogue Systems

April 23, 2022

**Candidate no.:** J902C

**Word count:** 2266<sup>1</sup>



---

<sup>1</sup>Excluding appendix, tables, footnotes, images, code and titles

# Table of contents

<b>1 Overview</b>	<b>1</b>
<b>2 Exercise GDBNMT.1</b>	<b>2</b>
2.1 Build a Word-to-Class Transducer . . . . .	2
2.2 Build a Gender Mapping Transducer . . . . .	3
<b>3 Exercise GDBNMT.2</b>	<b>4</b>
3.1 Build a Word-to-BPE Transducer . . . . .	4
<b>4 Exercise GDBNMT.3</b>	<b>6</b>
4.1 Gender-Inflected WFSAs for Rescoring with Debiased Models . . . . .	6
<b>5 Exercise GDBNMT.4</b>	<b>7</b>
5.1 Running the SGNMT decoder . . . . .	7
5.2 SLURM Rescoring script . . . . .	7
5.3 Rescoring of WMT'18 and WinoMT test sets . . . . .	8
<b>6 Discussion and Conclusion</b>	<b>9</b>

# 1. Overview

The goal of this practical is reducing gender bias in neural machine translation (NMT) as a domain adaptation problem following the article [1].

NLP system often show gender bias mainly because of the unbalance quality of training data, where sentences referring to men are more common than the ones using feminine forms. This is a serious problem to tackle for NMT when languages are gender-inflected, since translations are better for sentences involving men and for sentences containing stereotypical gender roles.

Saunders's ACL 2020 paper proposes to use a small but trustworthy (hardcrafted) dataset that could allow more efficient and effective gender debiasing by transfer learning than a larger and noisier dataset. Also, the article mentions that improvements on the gender-debiased task cause a reduction in translation quality due to catastrophic forgetting. To balance that, [1] investigates two regularised training procedures: Elastic Weight Consolidation (EWC), or in inference by a two-step lattice rescoring procedure. This practical focuses on the second procedure.

The aim is to adapt a baseline NMT models to a gender-balanced adaptation set with the aim of improving accuracy while carrying out a two-pass decoding procedure to prevent any degradation in overall translation quality.

The baseline is based on the Facebook-FAIR English-to-German translation systems developed for the 2018 Workshop on Machine Translation (WMT'18). Facebook-FAIR models is formed by an ensemble of six Transformers, and translation hypothesis is  $\hat{y} = \operatorname{argmax}_y \sum_{i=1}^6 P(y|x; \theta_i)$ . In this project we are going to use Fairseq/PyTorch models distributed by Facebook-FAIR that are already provided in the HPC directories. Note that these baseline models were built for the WMT'18, so their performance is already very high.

To measure the results of the systems investigated here, two metrics are going to be considered: WinoMT and BLEU. WinoMT provides a suite of tools for assessing the gender translation accuracy for systems. Also, the toolkit `sacrebleu` provides routines to measure BLEU score, and the WMT'18 English-German test set is a standard test supported by this tool.

## 2. Exercise GDBNMT.1

### 2.1. Build a Word-to-Class Transducer

The first part of this exercise has the goal of creating a *flower transducer* (`fsts/wtoc.fst`) that maps every word to itself and every word to all classes it belongs. To do so we use the file `$GDBNMTBDIR/wordclasses` that maps words to classes and the file `$GDBNMTBDIR/fsts/w+l.map.de` that maps words, classes and symbols to its encoding.

A python script is implemented to create a `wtoc.txt` with the following format:

```
0 0 word_1 word_1
0 0 word_1 class_word_1
...
0 0 <epsilon> <epsilon>
...
0
```

This describes an FST that maps every word to itself (including special characters) and to every class it belongs. This FST can be compiled using the shell command:

```
1 $ fstcompile --isymbols=$GDBNMTBDIR/fsts/w+l.map.de \
2 --osymbols=$GDBNMTBDIR/fsts/w+l.map.de --keep_isymbols \
3 --keep_osymbols fsts/wtoc.txt fsts/wtoc.fst
```

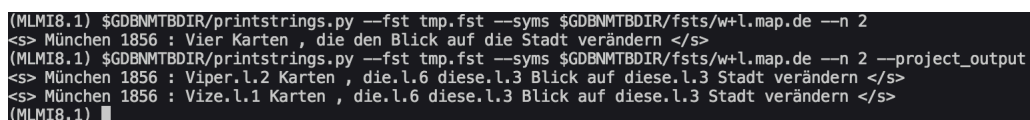
After creating `wtoc.fst` transducer, we can apply it to the acceptors in the directories:

```
- $GDBNMTBDIR/fsts/wmt18.sgnmt.wmt18ensemble.1/
- $GDBNMTBDIR/fsts/winomt.sgnmt.wmt18ensemble.1/
```

to create transducers whose input language is the baseline translation and whose output language contains all possible class mappings. To do that, we simply *compose* a transducer in any of the mentioned directories with `wtoc.fst`, as an example:

```
1 $ fstcompose $GDBNMTBDIR/fsts/wmt18.sgnmt.wmt18ensemble.1/1.fst \
2 fsts/wtoc.fst > tmp.fst
```

Then, the provided script `$GDBNMTBDIR/printstrings.py` can be employed to get all possible class mappings for a given text portion of the baseline translation, as exemplified in figure 2.1 where the word 'Vier' in the baseline translation is replaced by all its classes 'Vize.l.1' and 'Viper.l.2', the word 'die' is replaced by the class 'die.l.6', etc.



```
(MLMI8.1) $GDBNMTBDIR/printstrings.py --fst tmp.fst --syms $GDBNMTBDIR/fsts/w+l.map.de --n 2
<s> München 1856 : Vier Karten , die den Blick auf die Stadt verändern </s>
(MLMI8.1) $GDBNMTBDIR/printstrings.py --fst tmp.fst --syms $GDBNMTBDIR/fsts/w+l.map.de --n 2 --project_output
<s> München 1856 : Viper.l.2 Karten , die.l.6 diese.l.3 Blick auf diese.l.3 Stadt verändern </s>
<s> München 1856 : Vize.l.1 Karten , die.l.6 diese.l.3 Blick auf diese.l.3 Stadt verändern </s>
(MLMI8.1)
```

Figure 2.1: Baseline text words replaced with their possible classes.

## 2.2. Build a Gender Mapping Transducer

The goal now is to create a transducer `T.fst` that maps each word in the baseline translations to its gendered alternatives using only the transducer `wtoc.fst` previously created, as exemplified in fig. 2.2.

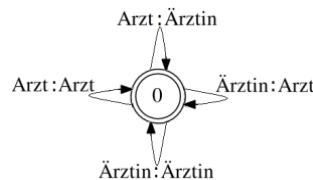


Figure 2.2: A subset of flower transducer `T.fst`, that maps vocabulary to itself as well as to differently-gendered inflections. Image from [1].

The transducer `wtoc.fst` maps every word to its class/classes, so to get all gendered versions of a given word we just have to get all words that are in the same class. To do so, we can invert the transducer `wtoc.fst` to get the mapping between classes to words. This transducer is going to be called `ctow.fst`. The inverted FST can be computed by:

```
1 $ fstinvert fsts/wtoc.fst fsts/ctow.fst
```

Now, both transducers can be composed to generate a transducer that maps a word to all its possible generated alternatives. Before composing, we have to sort both transducers' input labels so the composition can be done:

```
1 $ fstarcsort --sort_type=ilabel fsts/wtoc.fst fsts/wtoc.fst
2 $ fstarcsort --sort_type=ilabel fsts/ctow.fst fsts/ctow.fst
3
4 $ fstcompose fsts/wtoc.fst fsts/ctow.fst fsts/T.fst
```

The generated `T.fst` transducer can be composed again with the input baseline translation WFSA's to create a transducer that outputs all gendered versions of a given phrase. Using the same example code provided in the practical statement, we can check the correct implementation of transducer `T` (fig. 2.3).

```
(MLM18.1) $GDBNMTBDir/printstrings.py --fst tmp.fst --n 2 --syms $GDBNMTBDir/fsts/w+l.map.de
<s> München 1856 : Vier Karten , die den Blick auf die Stadt verändern </s>
(MLM18.1) $GDBNMTBDir/printstrings.py --fst tmp.fst --n 4 --syms $GDBNMTBDir/fsts/w+l.map.de --project_output
<s> München 1856 : Vize Katen , dieser diesen Blick auf dieser Stab verändern </s>
<s> München 1856 : Vize Karteien , dieser diesen Blick auf dieser Stabes verändern </s>
<s> München 1856 : Vikars Kauen , dieser diesen Blick auf die Stadt verändern </s>
<s> München 1856 : Vize Kauen , dieser diesen Blick auf dieser Stabes verändern </s>
```

Figure 2.3: Baseline text words replaced with their possible gendered alternatives.

## 3. Exercise GDBNMT.2

### 3.1. Build a Word-to-BPE Transducer

We are now interested in converting words into their corresponding Byte Pair Encoded (BPE) subwords. For example, the word 'Aachener' in the baseline should be converted to 'A@@ ach@@ ener'. To do so, a mapping for all the words in the test sets to their subword (BPE) form is provided in `$GDBNMTBDir/fairseq.pretrained/word_bpe.dict`.

To solve this problem, a flower transducer is built, called `wtobpe.fst` that maps word sequences to subwords sequences in their BPE form. Since this transducer needs to map a word to a sequence of subwords if the word has several subwords, the transducer has to have several states with `<epsilon>` as input and a subword as output. Figure 3.1 is a subset of `wtobpe` transducer, where the word 'Abbau' is mapped to itself since it only has one subword, and the word 'Abbaue' is mapped to the list of its subwords using auxiliary states.

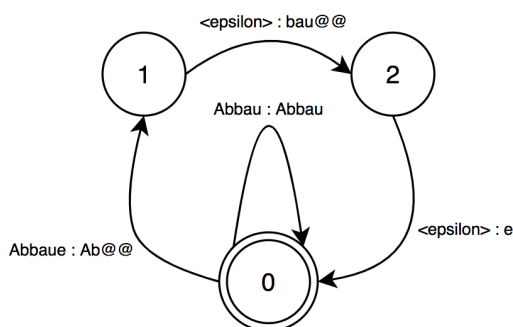


Figure 3.1: Subset of transducer that maps word sequences to subword sequences. Here, the word 'Abbau' is composed by only one subword 'Abbau', and in contrast, the word 'Abbaue' is composed by three subwords (BPE form): 'Ab@@', 'bau@@' and 'e'.

A file `wtobpe.txt` is generated using the `mappingword_bpe.dict` and it describes the transducer `wtobpe.fst`. A sample of the final `.txt` file is shown in figure 3.2.

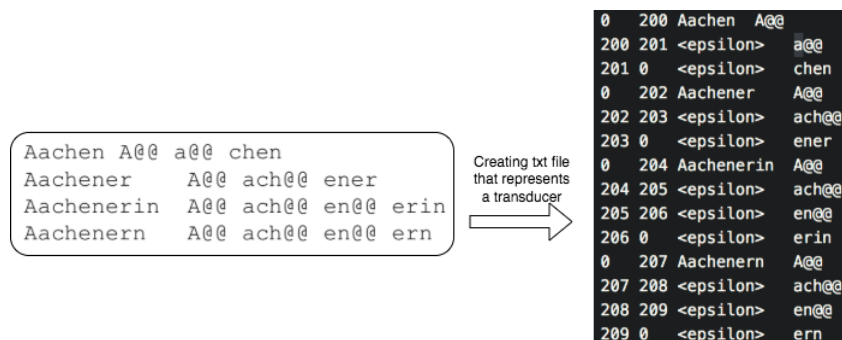


Figure 3.2: Fragment of the `.txt` file that represents flower transducer `wtobpe`.

Then, `wtobpe.txt` can be compiled to generate `wtobpe.fst`:

```
1 $ fstcompile --isymbols=$GDBNMTBDIR/fsts/w+l.map.de \  
2 --osymbols=$GDBNMTBDIR/fairseq.pretrained/wmap.bpe.de --keep_isymbols \  
3 --keep_osymbols fsts/wtobpe.txt fsts/wtobpe.fst
```

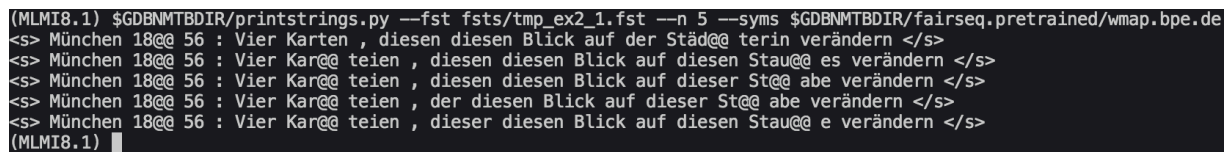
Before applying `wtobpe.fst` to any other transducer, we have to sort the arc labels, so they match other transducer's labels:

```
1 $ fstarcsort --sort_type=ilabel fsts/wtobpe.fst fsts/wtobpe.fst
```

Finally, transducers `T.fst` and `wtobpe.fst` can be composed (in this order) with an input baseline translation, followed by output projection, to get a WFSA that generates all gendered alternatives in their subword (BPE) form of the given input translation.

```
1 $ fstcompose $GDBNMTBDIR/fsts/wmt18.sgnmt.wmt18ensemble.1/1.fst \  
2 fsts/T.fst | fstcompose - fsts/wtobpe.fst | \  
3 fstproject --project_type=output > tmp.fst
```

Figure 3.3 shows some gendered alternatives in their subword (BPE) form for the input baseline translation `wmt18.sgnmt.wmt18ensemble.1/1`.



```
(MLMI8.1) $GDBNMTBDIR/printstrings.py --fst fsts/tmp_ex2_1.fst --n 5 --syms $GDBNMTBDIR/fairseq.pretrained/wmap.bpe.de  
<s> München 18@@ 56 : Vier Karten , diesen diesen Blick auf der Städ@@ terin verändern </s>  
<s> München 18@@ 56 : Vier Kar@@ teien , diesen diesen Blick auf diesen Stau@@ es verändern </s>  
<s> München 18@@ 56 : Vier Kar@@ teien , diesen diesen Blick auf dieser St@@ abe verändern </s>  
<s> München 18@@ 56 : Vier Kar@@ teien , der diesen Blick auf dieser St@@ abe verändern </s>  
<s> München 18@@ 56 : Vier Kar@@ teien , dieser diesen Blick auf diesen Stau@@ e verändern </s>  
(MLMI8.1)
```

Figure 3.3: Baseline text words replaced with their possible gendered alternatives in their subword form.



## 4. Exercise GDBNMT.3

Using the crafted transducers from previous exercises, `T.fst` and `wtobpe.fst`, a set of gender-inflected WFSAs (as acceptors) for rescoring can be created for both WMT'18 and WinoMT sets.

There are two additional things to take into account in this exercise. First, the automata should be optimized to make the posterior rescoring procedure faster. Optimization is done by determinizing and minimizing the created automatas. Determinization generates an equivalent automata that for each state and each symbol of the alphabet, there is always a transition. Minimization generates an equivalent minimal automata, i.e., an equivalent automata with the minimum number of possible states.

On the other hand, Fairseq/PyTorch libraries assume that the index 0 indicates a start-of-sentence marker, and OpenFST reserves this index for the `<epsilon>` symbol. To solve this conflict, the transducer `remapstartsym.fst` is provided to do this remapping of start symbols. It is important to do the remapping after all the optimizations are completed, otherwise OpenFST will treat the start-of-sentence symbol as an epsilon.

### 4.1. Gender-Inflected WFSAs for Rescoring with Debiased Models

To sum up, each baseline translation is composed with transducers `T.fst` and `wtobpe.fst`. To avoid inefficient edges with input and output symbol `<epsilon>` we prune the composed transducer, deleting these links and generating an equivalent FST. To generate acceptors, the FST is projected by its output. The executed commands to do the former are:

```
1  fstcompose $baseline_translation_file fsts/T.fst | \  
2  fstcompose - fsts/wtobpe.fst > fsts/tmp0.fst  
3  
4  fstrmepsilon fsts/tmp0.fst fsts/tmp0.fst  
5  fstproject --project_type=output fsts/tmp0.fst > fsts/tmp1.fst
```

After that, optimizations and start symbol remapping are executed using the following commands.

```
1  fstdeterminize fsts/tmp1.fst fsts/tmp1.fst  
2  fstminimize fsts/tmp1.fst fsts/tmp1.fst  
3  
4  output=fsts/wmt18.sgnmt.wmt18ensemble.1.ga/${i}.fst  
5  fstcompose fsts/tmp1.fst $GDBNMTBDIR/fsts/remapstartsym.fst |\  
6  fstproject --project_type=output > $output
```

This process is repeated for each baseline file of both WMT'18 and WinoMT sets. There are in total 2998 WMT'18 files and 3888 WinoMT files.

These acceptors can be tested as it was done in the previous sections, using provided python script `printstrings.py`. Figure 4.1 shows a example output where some words of the first file of WMT'18 set have been replaced with their gendered alternatives in their subword form.

```
(MLMI8.1) $GDBNMTBDir/printstrings.py --fst fsts/tmp.fst --syms $GDBNMTBDir/fairseq.pretrained/wmap.bpe.de --n 5
München 18@@ 56 : Vier Karten , diesen diesen Blick auf der Städ@@ terin verändern </s>
München 18@@ 56 : Vier Kar@@ teien , der diesen Blick auf dieser St@@ abe verändern </s>
München 18@@ 56 : Vier Kar@@ teien , diesen diesen Blick auf dieser St@@ abe verändern </s>
München 18@@ 56 : Vier Kar@@ teien , diesen diesen Blick auf diesen Stau@@ es verändern </s>
München 18@@ 56 : Vier Kar@@ teien , dieser diesen Blick auf diesen Stau@@ e verändern </s>
(MLMI8.1)
```

Figure 4.1: Baseline text words of file 1 of WMT'18 set are replaced with their possible gendered versions in subword form.

## 5. Exercise GDBNMT.4

The generated acceptors in exercise 3 (section 4) can be used for decoding and rescoring.

### 5.1. Running the SGNMT decoder

A provided python script `decode.py` can be executed to decode (translate) English sentences in their BPE form. In the following example, line 19 from the specified source file `$GDBNMTBDir/fairseq.pretrained/winomt.en-de.en.bpe` is translated.

```
1 SGNMT=/rds/project/rds-xyBFuSj0hm0/MLMI8.L2022/src/sgnmt.Nov21
2
3 python3 $SGNMT/decode.py \
4     --config_file=$GDBNMTBDir/configs/wmt18.1.ende.wfsa.adapt.1.ini \
5     --range=19:19 \
6     --output_path=tmp/winomt \
7     --src_test=$GDBNMTBDir/fairseq.pretrained/winomt.en-de.en.bpe \
8     --fst_path=fsts/winomt.sgnmt.wmt18ensemble.1.ga/%d.fst
```

It can be seen by comparing the output with the baseline translation that rescoring generates an alternative to the baseline translation. In this case, the generated translation of the English sentence "The physi@@ cian told the b@@ aker that she tried the best ." is the German sentence "Die Ärztin sagte der Bäckerin , dass sie alles versucht habe ." (both in BPE form).

### 5.2. SLURM Rescoring script

To translate the entire WinoMT and WMT18 sets, a SLURM decoding script is also provided, that can submit SGNMT rescoring routines to the HPC cluster. This is also useful since it can be easily parallelizable over multiple CPUs.

The script is executed by running:

```
1 sbatch slurm.decode.mjobs.cpu
```

Since the generated acceptors of exercise 3 (sec. 4) have been optimized and the script is parallelized, it will just take about 5 minutes to run, and the output translations are stored in `wmt18.0.01.90.cpu/` folder.

### 5.3. Rescoring of WMT'18 and WinoMT test sets

The output translations of the adapted models built in this practical can be scored using BLEU and WinoMT metrics and comparing the results to the ones of the baseline system.

BLEU score can be measured in WMT'18 output translations using the following command, and the results are shown in figure 5.1.

```
1 cat ./exps/wmt18.0.01.90.cpu/wmt18/wmt18.detok | \
2 sacrebleu --test-set wmt18 --language-pair en-de --format text
```

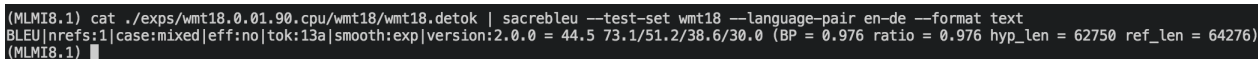
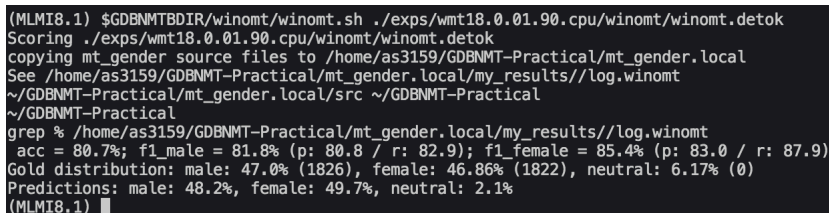


Figure 5.1: BLEU score of the WMT'18 output translations generated by the crafted adapted models in this practical.

The sacrebleu tool reports a BLEU score of 44.5 with 1-/2-/3-/4-gram precisions of 73.1/51.2/38.6/30.0 and a Brevity Penalty of 0.976. These results are really similar to the ones of the baseline system, where all the scores were identical but the 2-gram precision, that the baseline model got a 51.3 precision. This shows that the implemented adapted system in this practical is performing equally well as the baseline translations, that were generated with the SGNMT decoder and the first component model of the Facebook-FAIR ensemble.

On the other hand, WinoMT output translations can also be scored. The WinoMT scoring procedure is run as follows, and the results are illustrated in figure 5.2.

```
1 $GDBNMTBDir/winomt/winomt.sh ./exps/wmt18.0.01.90.cpu/winomt/winomt.detok
```



```
(MLM18.1) $GDBNMTBDir/winomt/winomt.sh ./exps/wmt18.0.01.90.cpu/winomt/winomt.detok
Scoring ./exps/wmt18.0.01.90.cpu/winomt/winomt.detok
copying mt_gender source files to /home/as3159/GDBNMT-Practical/mt_gender.local
See /home/as3159/GDBNMT-Practical/mt_gender.local/my_results//log.winomt
~/GDBNMT-Practical/mt_gender.local/src ~/GDBNMT-Practical
~/GDBNMT-Practical
grep % /home/as3159/GDBNMT-Practical/mt_gender.local/my_results//log.winomt
acc = 80.7%; f1_male = 81.8% (p: 80.8 / r: 82.9); f1_female = 85.4% (p: 83.0 / r: 87.9)
Gold distribution: male: 47.0% (1826), female: 46.86% (1822), neutral: 6.17% (0)
Predictions: male: 48.2%, female: 49.7%, neutral: 2.1%
(MLM18.1)
```

Figure 5.2: Accuracy of the WinoMT output translations generated by the crafted adapted models in this practical.

The scoring system reports an overall accuracy of 80.7%, and a male and female entity F1-scores of 81.8% and 85.4% respectively. The baseline system reported an overall score of 78.3%, with with male and female F1 entity scores of 79.5% and 82.8%. Therefore, the implemented model in this practical improves the baseline system performance in all the reported scores, so we demonstrate that bias can be further reduced without degradation in translation quality.

## 6. Discussion and Conclusion

In this practical a NMT system has been built with the aim of reducing gender bias while preventing any degradation in overall translation quality. The system is formed by the composition of several finite-state transducers and the generation of WFSAs (as acceptors) for rescoring.

In the last exercise (section 5) we evaluated the performance of the implemented system, resulting in a BLEU score of 44.5 (same as the baseline system) and a overall accuracy of 80.7% of the WinoMT output translations, compared to the baseline system performance of 79.5%. This means that the developed approaches in this practical have improved the overall performance, reducing gender bias while avoiding catastrophic forgetting.

The question now is whether the metrics used here are suitable for measuring syntactic gender accuracy. BLEU is a metric for automatically evaluating machine-translated text. The BLEU score tries to measure the similarity of the machine-translated text to a set of high quality reference translations. This metric focuses on general MT quality and it is usually well correlated with human judgement, however, there is no guarantee that an increase in BLEU score is an indicator of improved translation quality ([2]). The other employed metric, WinoMT, is used to estimate the gender-bias of an MT model in a fixed target language by: first, translating all sentences in WinoMT into the target language using the model; second, aligning the source and target translations; and finally extracting the gender of the target language translations using simple heuristics. To sum up, this process extracts the translated genders according to the given model for all the entities in WinoMT, which can be evaluated against the gold annotations provided in the original English dataset. The described process can introduce noise into the evaluation via wrong alignments or incorrect morphological analysis, but in section 3 of [3] it is shown that these errors are infrequent, so WinoMT ends up being a decent metric to use for measuring syntactic gender accuracy.

On the other hand, changing the target language might be a feature of interest in the future. We could implement a new system with a different target language just by obtaining another rich and small handcrafted dataset, and use all the knowledge gathered during this project to come up with a effective and efficient system that translates while reducing gender bias.

Other option could be the usage of the transducers and acceptors developed in this practical. To do so, a new transducer mapping from German to the new target language should be created and then it could be composed with the system implemented in this practical. The final system would try to reduce the gender bias in the English-to-German set of transducers, and it would focus on just translating between German to the new target language. This method would have several drawbacks. First, a considerably

large dataset should be obtained to map German to the new target language, something that is not usually easy. Additionally, the overall system could be ineffective depending on the number of gender inflections in the target language. German can have 3 or even more gender inflections, and Spanish, for example, just 2. This could be a difficulty to be handled by the final system.

## Bibliography

- [1] Danielle Saunders and Bill Byrne. (2020). 'Reducing Gender Bias in Neural Machine Translation as a Domain Adaptation Problem'. *Association for Computational Linguistics*. <https://aclanthology.org/2020.acl-main.690v2.pdf>
- [2] Chris Callison-Burch, Miles Osborne and Philipp Koehn. (2016). 'Re-evaluating the Role of BLEU in Machine Translation Research'. *Association for Computational Linguistics*. <https://www.cs.jhu.edu/~ccb/publications/re-evaluating-the-role-of-bleu-in-mt-research.pdf>
- [3] Gabriel Stanovsky, Noah A. Smith and Luke Zettlemoyer (2019). 'Evaluating Gender Bias in Machine Translation'. *Association for Computational Linguistics*. <https://aclanthology.org/P19-1164.pdf>

# Task Oriented Dialogue State Tracking with GPT-2

MLMI 8 - Neural Machine Translation and  
Dialogue Systems

April 23, 2022

**Candidate no.:** J902C

**Word count:** 2320<sup>1</sup>



---

<sup>1</sup>Excluding appendix, tables, footnotes, images, code and titles

# Table of contents

<b>1 Overview</b>	<b>1</b>
<b>2 Exercise GPT2DST.1</b>	<b>2</b>
<b>3 Exercise GPT2DST.2</b>	<b>3</b>
<b>4 Exercise GPT2DST.3</b>	<b>4</b>
<b>5 Exercise GPT2DST.4</b>	<b>5</b>
5.1 Building our own GPT-2 DST system . . . . .	5
5.2 Comparison to UBAR . . . . .	7
5.3 Model Card of our own GPT-2 DST system . . . . .	8
<b>6 Summary and Conclusion</b>	<b>10</b>



# 1. Overview

In this practical Natural Language Understanding (NLU) and Dialogue State Tracking (DST) are investigated using GPT-2 transformer models [1].

Natural Language Understanding is the automatic annotation of dialogue acts within individual utterances or dialogue turns. NLU is performed and evaluated at the turn level within each dialogue. On the other hand, Dialogue State Tracking consists of determining at each turn of a dialogue the full representation of what the user wants at that point in the dialogue.

Transformers are usually language models trained on a huge text collection and they are described by a large amount of parameters. Training a transformer model is unfeasible for the computational power available, but a pre-trained model (GPT-2 model from HuggingFace [1]) can be downloaded and fine-tuned with relatively small amount of in-domain data. One of the keys to successfully develop this project is to determine how the fine-tuning task is formulated and executed.

The Multi-Domain Wizard-of-Oz (MultiWOZ) data sets are going to be used to report performance of in Natural Language Understanding and Dialogue State Tracking after fine-tuning several GPT-2 models on the in-domain data. MultiWOZ is a collection of human-human written conversations over multiple topics and it is widely used in the literature for reporting progress in DST. There exist different versions, because the dataset has been cleaned several times. Here, in this practical, the results will be reported just in the MultiWOZ 2.1 version [4].

	Train	Dev	Test
Turns	54971	7374	7368
Dialogues	7888	1000	999

Table 1.1: MultiWOZ 2.1 training, development, and test set sizes in this practical.

Exercises 1 and 2 (sections 2 and 3 respectively) will focus on fine-tuning, decoding and scoring for NLU. Exercise 3 (section 4) will consist on DST reporting the joint accuracy of the investigated models and, finally, in exercise 4 (section 5) we will build our own GPT-2 DST system.

## 2. Exercise GPT2DST.1

The very first step and the first exercise is to fine-tune a pre-trained GPT-2 model for NLU. The GPT-2 model is pre-trained in vast datasets, but the fine-tuning is done just on the MultiWOZ 2.1 dataset. We are provided with the dataset in the expected format to fine-tune the GPT-2 model.

A script named `train.nlu.mwoz21.slurm.wilkes2` is provided to fine-tune a HuggingFace GPT-2 model. The script can be set to train from models provided at batch 60000 or from scratch, i.e., a flat-start fine tuning from GPT-2 as distributed by HuggingFace [1]. In our case, we decide to execute the first option and keep fine-tuning the GPT-2 model based on the fine-tuned model until batch 60000 (continued training). To do so, we just have to submit a SLURM job to the server using the following command:

```
1 sbatch train.nlu.mwoz21.slurm.wilkes2
```

After approximately 2 hours the script is done and results are shown in table 2.1. Losses (negative log-likelihood) are reported in both training and developing sets. Results until batch 60000 are collected from the provided log at `$BDIR/checkpoints/nlu_flat_start/logs/train-gpt2-dst.log`. The rest of the results report development set loss every 20,000 batches, up to 200,000 batches, and training set loss every epoch.

Batch	Train loss	Dev loss
20000	0.4144	0.4037
40000		0.2639
54971		0.2180
60000		0.2093
80000		0.2323
100000	0.2325	0.2031
114971		0.2112
120000		0.2254
140000		0.2088
160000		0.2285
169942	0.1916	0.2105
180000		0.2454
200000		0.2866

Table 2.1: Replication of MultiWOZ 2.1 NLU executing fine-tuning as continued training. Development set loss is reported every 20000 turns and training loss is reported every epoch (54,971 turns).

Comparing table 2.1 with the one provided in the practical statement, we can analyze results after batch 60000 and see that they are pretty similar. Comparison can be

done just up to batch 120000 because the provided table is incomplete, but analyzing batch 80K and 100K we can see that dev. losses are in the same page. In general, the model tend to improve its performance and it is *expected* that continued training performs better in test set after 200K batches than just fine-tuning the model up to 60K batches.

### 3. Exercise GPT2DST.2

After fine-tuning HuggingFace GPT-2 models on MultiWOZ 2.1 NLU training sets, a turn level belief state annotation can be generated for each dialogue turn in the input file by decoding.

It is provided another SLURM script `decode.nlu.mwoz21.slurm.wilkes2` to run the decoder over the data. The script has to be modified to specify the fine-tuned GPT-2 model path, including the line:

```
1 CHECKPOINT=/rds/user/as3159/hpc-work/checkpoints/nlu_continued_training  
/model.80000/
```

In the previos example, the variable `CHECKPOINT` stores the path where a GPT-2 model has been fine-tuned with 80000 batches. The script can be submitted to the HPC servers by executing the command:

```
1 sbatch decode.nlu.mwoz21.slurm.wilkes2
```

The output of this script includes the dialogue turn itself, copied directly from the input field `dst_input`, and the predicted linearized belief state follows the input, appearing between the `<BOS>` and `<EOS>` tokens.

After decoding, the provided python script `multiwoz_dst_score.py` can be executed to score the NLU decoder output against the reference turn level belief states. Table 3.1 shows the DST Average Joint Accuracy after fine-tuning a GPT-2 model by continued training. The results are reported following the practical statement, focusing from batch 60000 to 200000 every 20000 batches.

Batch	Dev	Test
5000	68.66	66.30
52000	70.56	67.44
54000	70.90	69.46
60000	69.24	67.13
80000	71.94	70.14
100000	70.22	68.59
120000	71.74	69.84
140000	72.20	70.28

160000	73.27	71.20
180000	73.07	71.85
200000	74.48	72.58

Table 3.1: NLU Joint Accuracy for MultiWOZ 2.1 found via decoding with GPT2NLU using continued training models

In general, DST Average Joint Accuracy in the development set tends to increase. Similarly, it also gradually increases on the test set, what means the GPT-2 is improving while being fine-tuned up to 200000 batches, reaching a top DST Average Joint Accuracy of 72.58%.

## 4. Exercise GPT2DST.3

The investigation of turn level Natural Language Understanding is extended to Dialogue State Tracking. As the base case, DST Joint Accuracy of the turn level annotations against the dialogue level reference can be measured, but it is expected that this accuracy will be considerably low because of the fact that the predicted belief state is based only on the current turn. This can be improved by simply concatenating turn level NLU belief state predictions, implementing a Cheap and Cheerful (*something that does not cost a lot but is attractive and pleasant*) Dialogue State Tracker.

The provided python script `cc_dst.py` concatenates the turn level belief states to generate dialogue level belief states. It is expected that the overall performance is limited, since it depends on the independence of the generated turn level annotations.

Once the dialogue level annotations have been generated, they can be scored, i.e., the DST Joint Accuracy can be measured. The following command would score the fine-tuned GPT2NLU CC-DST model by continued training (200000 batches) contrasting its annotations with the reference:

```
1 python $BDIR/multiwoz_dst_score.py --field dst_belief_state \
2 --dst_ref $BDIR/data_preparation/data/multiwoz21/refs/test/test_v2.1.
  json \
3 --h hyps/test/cc_dst/nlu_continued_training/model.200000/belief_states.
  json
```

Using this command on the development and test reference, the DST Joint Accuracy can be obtained for different GPT-2 models that have been fine-tuned for different number of batches. The results are shown in table 4.1.

Batch	Dev	Test
5000	34.26	31.46
52000	39.03	34.22
54000	41.36	37.76
60000	35.72	31.79
80000	41.69	38.57
100000	37.75	34.73
120000	40.29	37.19
140000	41.13	36.92
160000	42.26	39.45
180000	44.78	41.26
200000	46.46	42.55

Table 4.1: MultiWOZ 2.1 DST Joint Accuracy: GPT2NLU (continued training) and Cheap and Cheerful DST

In general, DST Joint Accuracy tend to increase on the development set, reaching a top accuracy of 46.46%. Focusing on the test set, the behaviour is similar. The DST Joint Accuracy usually improves if the GPT-2 model is fine-tuned for more batches. The best result on the test set, 42.55% DST Joint Accuracy, is obtained after fine-tuning for 200000 batches.

## 5. Exercise GPT2DST.4

### 5.1. Building our own GPT-2 DST system

So far we have investigated the models described in the statement, but now we are interested in designing, building and evaluating our own GPT-2 Dialogue State Tracker on the MultiWOZ 2.1 dataset.

The proposed model in this section merges NLU and DST into a single system. To do so, a modified version of the given dataset (turn level MultiWOZ 2.1) has to be built. From the turn level data provided a dialogue level dataset is crafted using a similar script to `cc_dst.py` (used in exercise 3, section 4). This script was a simple implementation of hypothesis concatenation, but in this exercise this script is extended to concatenate training, development and testing data.

The snippet 5.1 shows a fragment of the new dialogue level training dataset, where a turn contains information about the current turn itself and the previous part of the dialogue between the user and the system. This format is logically also used for development and testing data.

```
1  [
2      {
3          "example_id": "MUL0003-0",
4          "dst_input": "<USR> i am looking for a place to stay . it needs
5                      to be a guesthouse and include free wifi . "
6      },
7      {
8          "example_id": "MUL0003-1",
9          "dst_input": "<USR> i am looking for a place to stay . it needs
10                     to be a guesthouse and include free wifi . <SYS> there are
11                     23 hotel -s that meet your needs . would you like to narrow
12                     your search by area and and or price range ? <USR> i would
13                     like for it to be cheap and include free parking . "
14      },
15      {
16          "example_id": "MUL0003-2",
17          "dst_input": "<USR> i am looking for a place to stay . it needs
18                     to be a guesthouse and include free wifi . <SYS> there are
19                     23 hotel -s that meet your needs . would you like to narrow
20                     your search by area and and or price range ? <USR> i would
21                     like for it to be cheap and include free parking . <SYS>
22                     there are 9 guesthouse hotel -s in various area -s . what
23                     part of town are you hoping for ? <USR> nothing in particular
24                     . i just need it booked for 6 people for a total of 4 nights
25                     starting from sunday . i would also like the reference number
26                     , please . "
27      },
28  ]
```

Listing 5.1: Example of the new dialogue level data

The system is trained using the new dataset and executing a modified version of the `train.nlu.mwoz21.slurm.wilkes2` script, setting the variables `TRAIN_DATA` and `DEV_DATA` to the new training set and development set respectively. In this case, the model is fine-tuned from scratch (from GPT-2 as distributed by HuggingFace [1]) up to 200000 batches. After training, decoding can be run using a modified version of the script `decode.nlu.mwoz21.slurm.wilkes2`, where the trained model and the directory for the decoded dialogues are specified. Finally, scoring (on the test set) can be done by executing:

```
1  echo "Development set DST Joint Accuracy:"
2  python $BDIR/multiwoz_dst_score.py --field dst_belief_state \
3  --dst_ref $BDIR/data_preparation/data/multiwoz21/refs/dev/dev_v2.1.json \
4  --h ./hyps/ex4/dev/nlu_continued_training/model.200000/belief_states.
5  json
6  echo "Test set DST Joint Accuracy:"
7  python $BDIR/multiwoz_dst_score.py --field dst_belief_state \
8  --dst_ref $BDIR/data_preparation/data/multiwoz21/refs/test/test_v2.1.
9  json \
10 --h ./hyps/ex4/test/nlu_continued_training/model.200000/belief_states.
11 json
```

The results are shown in figure 5.1 and summarized in table 4.

```
(MLMI8.GPT2DST) [as3159@login-e-3 hpc-work]$ source ex4_score_dst.sh
Development set DST Joint Accuracy:
References: /rds/project/rds-xyBFuSj0hm0/MLMI8.L2022/GPT2DST//data_preparation/data/multiwoz21/refs/dev/dev_v2.1.json reference field: dst_belief_state
Hypotheses: ./hyps/ex4/dev/nlu_continued_training/model.200000/belief_states.json
DST Average Slot Accuracy: 97.19 %. DST Average Joint Accuracy: 51.40 %

Test set DST Joint Accuracy:
References: /rds/project/rds-xyBFuSj0hm0/MLMI8.L2022/GPT2DST//data_preparation/data/multiwoz21/refs/test/test_v2.1.json reference field: dst_belief_state
Hypotheses: ./hyps/ex4/test/nlu_continued_training/model.200000/belief_states.json
DST Average Slot Accuracy: 96.86 %. DST Average Joint Accuracy: 47.72 %
(MLMI8.GPT2DST) [as3159@login-e-3 hpc-work]$
```

Figure 5.1: Terminal output after scoring our own GPT-2 DST model, showing DST Average Joint Accuracy

Model	Batch	Dev	Test
CC-DST	200000	46.46	42.55
NLU+DST	200000	51.40	47.72

Table 5.1: Comparison of Cheap and Cheerful DST system (detailed in section 4) and the new proposed model that merges NLU and DST into a dialogue level system.

It is easy to check that the proposed system improves the results of the previous exercise (section 4), achieving a 47.72% DST Average Joint Accuracy on the test set. This was expected since in exercise 3 dialogue level belief states were just generated by concatenating the predictions, whether in this exercise the DST procedure was refined by building new dialogue level dataset and training, decoding and scoring using these enhanced data.

## 5.2. Comparison to UBAR

UBAR is a end-to-end system with GPT-2 ([3]) that models task-oriented dialogues on a dialog session level. UBAR is trained and evaluated fine-tuning a GPT-2 model in a more realistic setting than our proposed model in this section 5, since its dialog context has access to user utterances and all content it generated such as belief states, system acts, and system responses of every dialog turn. In contrast, the proposed NLU+DST system in this practical only uses dialogue level belief states for training and testing.

UBAR makes the process of inference easier for the current turn by conditioning on the previous belief states and system acts in the dialogue. Figure 5.2 shows an example of two-turn interaction and what information is UBAR processing.

The creators of UBAR also conducted experiments on the MultiWOZ 2.1 dataset, achieving state-of-the-art performances accordingly to [3]. This is perfect to compare the capability of UBAR with our system of section 5.1. The performances are compared in table 5.2.



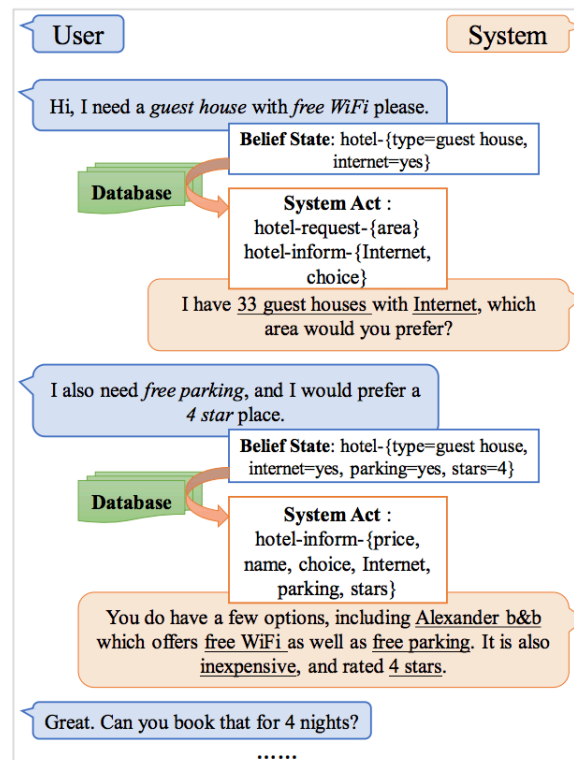


Figure 5.2: An example of the first two-turn interactions between a user and a TOD system. UBAR uses all this information for training and inference, whether the proposed system in 5.1 only uses dialogue level belief states. This image is sourced from [3].

Model	Dev	Test
NLU+DST	51.40	47.72
UBAR	-	56.20

Table 5.2: Comparison of UBAR and the new proposed model that merges NLU and DST into a dialogue level system. Units are measured in DST Average Joint Accuracy.

As it can be seen, UBAR outperforms our system, reaching a 56.20% DST Average Joint Accuracy on the test set, compared to the 47.72% of the proposed system in section 5.1. This was expected since UBAR uses much more information than the trained model in this project.

### 5.3. Model Card of our own GPT-2 DST system

The following model card describes our own GPT-2 DST system described in section 5.1. There are several formats for a model card, but here it is used the one proposed in [2].



**Model Details:**

- Built by 2022 MPhil. MLMI Cambridge student, using scripts elaborated by professor Bill Byrne to fine-tune and evaluate GPT-2 models.
- Dialogue State Tracking system with GPT-2. HuggingFace [1] GPT-2 transformer models are employed.
- Pre-trained in a huge general dataset and fine-tuned in this practical for task-oriented dialogue state tracking.
- LICENSE: MIT license.

**Intended Use:**

- Automatic annotation of dialogue acts within dialogue turns.
- Determining at each turn of a dialogue the full representation of what the user wants at that points in the dialogue.
- Not intended to be deployed to a final system, since it has several limitations.

**Metrics:**

- Performance reported using DST Average Joint Accuracy of the dialogue level annotations.

**Training Data:**

- Dialogue level belief states of several human-system interactions.
- MultiWOZ 2.1 training set [4].

**Evaluation Data:**

- Dialogue level belief states of several human-system interactions.
- MultiWOZ 2.1 test set [4].

**Ethical Considerations:**

- Dialogue annotations, which span 7 distinct task-oriented domains, are collected and publicly published by Amazon and Google researchers [4] to evaluate different DST systems.

**Caveats and Recommendations:**

- Does not use other sources of information, such as system acts or system responses.
- To improve the performance is worth checking UBAR system [3].

- Synthetic data has a limited range of domains. It is worth evaluating its performance in real-word situations.

#### **Quantitative Analyses:**

- Achieved 51.40% DST Average Joint Accuracy on the development set and 47.72% DST Average Joint Accuracy on the test set.
- The system out-performed previous investigated models, such as a CC-DST system based on concatenating turn level annotations to generate dialogue level predictions.
- There is room for improvements (e.g. UBAR system [3]).

## **6. Summary and Conclusion**

In this practical Natural Language Understand (NLU) and Dialogue State Tracking (DST) were investigated using provided GPT-2 transformer models by HuggingFace [1]. These transformer models were fine-tuned on the MultiWOZ 2.1 dataset [4] in order to evaluate performance in DST.

In the first 2 exercises we fine-tuned, decoded and scored GPT-2 systems for NLU. The results were quite similar to the ones provided in the practical statement as an example. The little variations in number could be due to randomness in initialization or optimization. In addition, exercise 3 consisted on DST, concatenating turn level belief states of the predictions to generate dialogue level belief states. The overall performance was limited since it depends on the independence of the generated turn level annotations.

The system investigated in exercise 3 got a 42.55% DST Average Joint Accuracy on the test set, and in exercise 4 we tried to improve this result by further investigating on DST. A new dataset was crafted in order to fine-tune a GPT-2 system using train, development and test data using dialogue level annotations. This approach achieved a final DST Joint Accuracy of 47.72%, outperforming the base result of exercise 3.

Future work could consist on improving the current top performing model including more information to use for training and inference or taking inspiration from other DST models, such as UBAR [3].

To sum up, in this practical we have tried to automatically annotate dialogue acts within dialogue utterances and also how to determine at each turn of a dialogue the full representation of what the user wants at that exact moment. The achieved results were not outstanding, but in the same page as other state-of-the-art models results.

## Bibliography

- [1] Thomas Wolf et. al. (2019). 'HuggingFace's Transformers: State-of-the-art Natural Language Processing'. *Computing Research Repository (CoRR)*. <http://arxiv.org/abs/1910.03771>
- [2] Margaret Mitchell et. al. (2019). 'Model Cards for Model Reporting'. *Association for Computing Machinery (ACM)*. <https://arxiv.org/abs/1810.03993>
- [3] Yunyi Yang et. al. (2021). 'UBAR: Towards Fully End-to-End Task-Oriented Dialog Systems with GPT-2'. *Computing Research Repository (CoRR)*. <https://arxiv.org/abs/2012.03539>
- [4] Mihail Eric et. al. (2019). 'MultiWOZ 2.1: Multi-Domain Dialogue State Corrections and State Tracking Baselines'. *Computing Research Repository (CoRR)*. <https://arxiv.org/abs/1907.01669>