

# 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
```

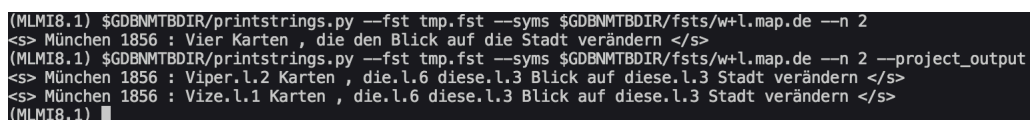
After creating `wtoc.fst` transducer, we can apply it to the acceptors in the directories:

```
- $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.

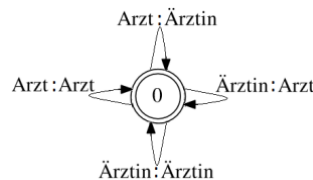


Figure 2.2: A subset of flower transducer `T.fst`, that maps vocabulary to itself as well as to differently-gendered inflections. Image from [1].

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 fstswtoc.fst fstswctow.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 fstswtoc.fst fstswtoc.fst
2 $ fstarcsort --sort_type=ilabel fstswctow.fst fstswctow.fst
3
4 $ fstcompose fstswtoc.fst fstswctow.fst fstswT.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/fstsw+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/fstsw+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.

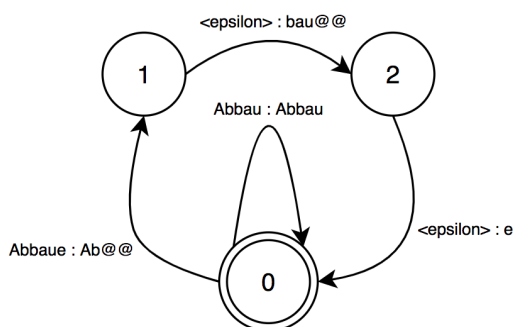


Figure 3.1: Subset of transducer that maps word sequences to subword sequences. Here, the word 'Abbau' is composed by only one subword 'Abbau', and in contrast, the word 'Abbaue' is composed by three subwords (BPE form): 'Ab@@', 'bau@@@' and 'e'.

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.

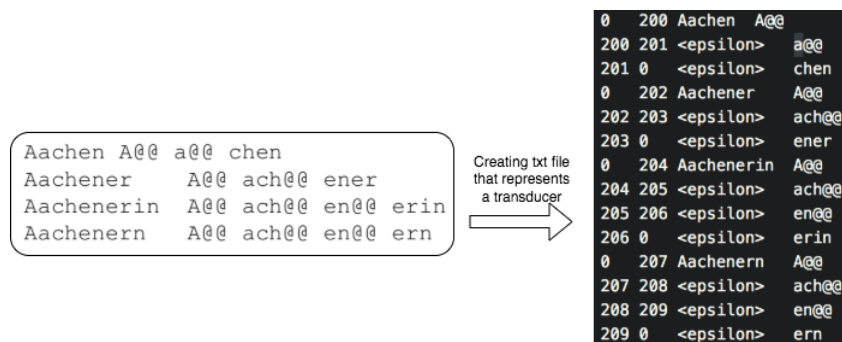


Figure 3.2: Fragment of the `.txt` file that represents flower transducer `wtobpe`.

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
```

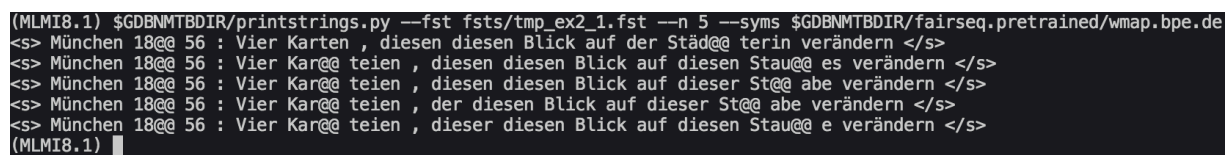
Before applying `wtobpe.fst` to any other transducer, we have to sort the arc labels, so they match other transducer's labels:

```
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 St@d@ 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
```

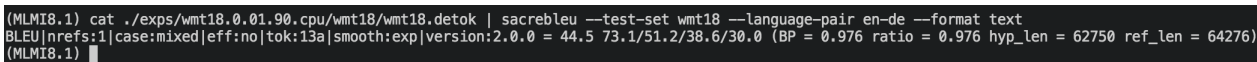
Since the generated acceptors of exercise 3 (sec. 4) have been optimized and the script is parallelized, it will just take about 5 minutes to run, and the output translations are stored in `wmt18.0.01.90.cpu/` folder.

### 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
```



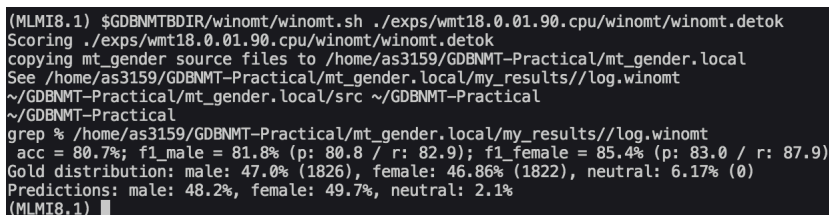
```
(MLM18.1) cat ./exps/wmt18.0.01.90.cpu/wmt18/wmt18.detok | sacrebleu --test-set wmt18 --language-pair en-de --format text
BLEU[nrefs:1|case:mixed|eff:no|tok:13a|smooth:exp|version:2.0.0 = 44.5 73.1/51.2/38.6/30.0 (BP = 0.976 ratio = 0.976 hyp_len = 62750 ref_len = 64276)
(MLM18.1) █
```

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 rescoreing.

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>