

Final Project Report
ON
Machine Translation of Noisy Text (MTNT)
Stevens Institute of Technology



STEVENS
INSTITUTE *of* TECHNOLOGY
THE INNOVATION UNIVERSITY®

Project Guide:

Professor Abdul Rafae Khan

Submitted by:

Janmejay Mohanty

jmohanty@stevens.edu

Course Name:

CS 541-A Artificial Intelligence

Abstract

In most modern Machine Translation (MT) systems, noisy or non-standard input text can result in devastating mistranslations, and there has been a growing research interest in developing noise-resistant MT systems. However, there are currently no publicly available parallel corpora of naturally occurring noisy inputs and translations, therefore past research has had to rely on synthetically constructed datasets for evaluation. The author presents a benchmark dataset for Machine Translation of Noisy Text (MTNT) in this study, which includes noisy Reddit comments and professionally supplied translations. On the huge number of sentences per language pair, we commissioned translations of English comments into French and Japanese, as well as French and Japanese comments into English. We look at the many types of noise in this dataset both qualitatively and quantitatively.

Some sample analysed data stats given as below:

1. Counted the number of profanities using the following below command:
`cat MTNT/test/test.en-fr.en | python3 analysis/count_keywords.py resources/profanities.en`
Result = 38
2. Counted the number of emojis (after importing the emojis module in python) using the following below command:
`cat MTNT/test/test.en-fr.en | python3 analysis/count_emojis.py`
Result = 46
3. Checking the ratio between US/UK spelling (for ise/ize which is a good indicator) using the following below command:
`cat MTNT/test/test.en-fr.en | python3 analysis/uk_us_ratio.py`
Result = 35.7% and 64.3%

Git-hub Link:

https://github.com/Janmejaya1998/MTNT_AI

Introduction

There are many random reddit messages and text data that contains large noisy data and also contains slangs.

#nlproc is actually f*ing hARD tbh

The above reddit text is an example of containing slangs and noisy data.

Sometimes it also contains emojis. Although the machine translation algorithm has been improved quite in the past few years due to many introductions of Neural Network and Machine Learning algorithms. But still there are many human errors in social media text which are still not being able to get resolved by these new coming Neural network and machine learning algorithms.

So right now, our current goal is to perform test on standard translation models and language models on our data for understanding their failure cases and to provide better and optimized approaches for the future task.

We faced many problems such as many deprecated lines of code, outdated and inappropriate libraries path and files directory which we had fixed and resolved it but still there are some error and problems remains which we had to fix it but unable to do so.

Data

For each different language, we are selecting a set of groups (“subreddits”) that we know contains many comments in that language such as English: Since a huge majority of the discussions on Reddit took place in the medium of English, so we don’t restrict our collection to any community in particular.

English: As huge majority of the discussions on Reddit are conducted in English, we are not restricting our collection to any specific community.

French: The total of the French side of the parallel training data which is provided for the English-French WMT 2015 translation procedure. This amount is to be approximately 40.86 million sentences.

Japanese: We had collected three small/medium sized MT datasets: KFTT (Neubig, 2011), JESC (Pryzant et al.) and TED talks (Cettolo et al., 2012), amounting to be approximately ≈ 4.19 million sentences.

The data files for MTNT can be found in this link:

https://github.com/Janmejaya1998/MTNT_AI/tree/main/mtnt-master/config

Most of the data we are dealing with are from Reddit based data.

Tools & Technologies

Software Requirements:

Currently running on Windows Operating System:

Version = Windows 10 (Latest)

Using both version of python 2.7 and 3.6

To run this project code, you are required the following python modules to be installed:

1. kenlm
2. langid
3. numpy
4. pickle
5. praw
6. sentencepiece >=0.1.6 (Version)
7. yaml

Also, you are required to use Ubuntu Terminal as it contains both shell codes and programs:

Ubuntu Version = 18.04 LTS

Using VirtualBox 6.1 tool to create and setup Ubuntu Terminal version 18.04 LTS for running the project.

Hardware Requirements:

Using personal CPU and GPU which are:

1. CPU = AMD RYZEN 7 5800H
2. GPU = NVIDIA GEFORCE RTX 3060

Data

For pre-processing the data, you are required to perform following things:

1. Moses: For tokenization, clean-up, etc... (<https://github.com/moses-smt/mosesdecoder>)
2. Sentencepiece: For subwords. (<https://github.com/google/sentencepiece>)
3. KenLM: For n-gram language modelling. (<https://kheafield.com/code/kenlm/>)

If you want to work on Japanese language data you can install Kytea (for word segmentation)

Reference link: <http://www.phontron.com/kytea/>

For preparing and downloading the data use the given below commands:

Monolingual en data from WMT17

```
bash scripts/download_en.sh config/data.en.config
```

```
bash scripts/prepare_model config/data.en.config
```

Monolingual fr data from WMT15

```
bash scripts/download_fr.sh config/data.fr.config
```

```
bash scripts/prepare_model config/data.fr.config
```

Prepare en<->fr parallel data

```
bash scripts/prepare-en-fr.sh config/data.fr.config path/to/moses/scripts
```

Download and prepare the en<->ja monolingual and parallel data

```
bash scripts/download_ja.sh config/data.ja.config path/to/moses/scripts
```

New MTNT File uploaded here

https://github.com/Janmejaya1998/MTNT_AI/tree/main/mtnt-master/MTNT

Split the tsv files

```
bash MTNT/split_tsv.sh
```

After completing the above command execution, for Running the Scraper you can go through the below description and some commands are mentioned below:

For editing the config/{en,fr,ja}_reddit.yaml to include the appropriate credentials for your own bot. You can modify some of the parameters and settings such as subreddits, etc...

For this you have to run this command which I have mention below:

```
bash scripts/start_scraper.sh [config_file]
```

Note: When you are running this scraper, please go through the Reddit API terms on their official website. (<https://www.reddit.com/wiki/api>)

After completing all these procedures, for analysis the data, please use the following given below commands for project execution:

```
# Count the number of profanities (should return 38)
```

```
cat MTNT/test/test.en-fr.en | python3 analysis/count_keywords.py resources/profanities.en
```

```
# Count the number of emojis (should return 46)
```

```
cat MTNT/test/test.en-fr.en | python3 analysis/count_emojis.py
```

```
# Check the ration US/UK spelling (for ise/ize which is a good indicator) (should return 35.7% 64.3%)
```

```
cat MTNT/test/test.en-fr.en | python3 analysis/uk_us_ratio.py
```

```
# Count the number of informal pronouns (in japanese) (should return 35)
```

```
kytea -model /path/to/kytea/data/model.bin -out tok MTNT/test/test.ja-en.ja | python3  
analysis/count_keywords.py resources/informal_pronouns.ja
```

Results:

This shows the output results of the files being generated for MTNT.

```
Activities Terminal
File Edit View Search Terminal Help
jraye@jraye-VirtualBox: ~/Documents/mtnt-master
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=10885 size=500 all=12817 active=4075 piece=_f
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=10865 size=500 all=12336 active=10198 piece=_system
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=10769 size=500 all=12595 active=11837 piece=_bas
bpe_model_trainer.cc(187) LOG(INFO) updating active symbols. max_freq=14015 min_freq=119
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=12488 size=520 all=14818 active=5151 piece=_ble
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=12584 size=568 all=15923 active=6664 piece=_act
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=11932 size=568 all=16137 active=8878 piece=IId
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=11537 size=500 all=16398 active=10713 piece=_ule
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=12080 size=400 all=15244 active=11895 piece=_lit
bpe_model_trainer.cc(187) LOG(INFO) updating active symbols. max_freq=18407 min_freq=1979
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=18517 size=478 all=16885 active=4923 piece=_unl
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=16932 size=608 all=16977 active=7937 piece=IId
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=17164 size=608 all=17164 active=10687 piece=oving
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=16422 size=700 all=17393 active=11861 piece=_fol
bpe_model_trainer.cc(187) LOG(INFO) updating active symbols. max_freq=18407 min_freq=1979
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=16471 size=770 all=17477 active=4518 piece=_area
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=1776 size=140 all=17616 active=6288 piece=_u
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=16955 size=760 all=17637 active=7878 piece=_al
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=16278 size=900 all=17794 active=10568 piece=_with
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=16077 size=900 all=18104 active=11847 piece=ment
bpe_model_trainer.cc(187) LOG(INFO) updating active symbols. max_freq=18407 min_freq=1979
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=17296 size=120 all=18281 active=1546 piece=_possible
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=17542 size=440 all=18783 active=6716 piece=_indust
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=16126 size=440 all=18981 active=8376 piece=_ince
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=17219 size=480 all=19724 active=9999 piece=ability
bpe_model_trainer.cc(187) LOG(INFO) updating active symbols. max_freq=18407 min_freq=1979
bpe_model_trainer.cc(258) LOG(INFO) Added: freq=17026 size=520 all=19212 active=10548 piece=_m
trainer_interface.cc(615) LOG(INFO) Saving model: model/corpus.en.1000.bpe.model
trainer_interface.cc(616) LOG(INFO) Saving vocab: model/corpus.en.1000.bpe.vocab
tokenizing
Training LM
=== 1/3 Counting and sorting n-grams ===
Reading /home/jraye/Documents/mtnt-master/jmt/corpus.en.1000.bpe.txt
...5...18...11...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95...100
=====
Unigram tokens 20811081 types 11170
=== 2/3 Calculating and sorting adjusted counts ===
Chara sizes: 1134840 1180771152 5112600318 4208263584 5379548532
Statistics:
1 11378 02=0.886472 02=1.18860 03=1.49841
2 49912 01=0.12322 01=0.894383 03=1.36233
3 1138065 01=0.837807 01=1.60647 01=1.51776
4 48878788 01=0.785157 02=1.11882 01=1.43818
5 9076413 01=0.752494 02=1.10282 01=1.43033
Memory estimate for binary LM:
type
probing 2937 assuming -p 1.5
probing 3274 assuming n models p 1.5
trie 1224 without quantization
trie 184 assuming -p 1.5 quantization
trie 1894 assuming -p 22 array pointer compression and quantization
trie 184 assuming -p 22 array pointer compression and quantization
=== 3/3 Calculating and sorting initial probabilities ===
Chara sizes: 1134840 117980000 3122085300 4112744264 51239712064
=====
5 18 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
=====
6/5 Calculating and writing order-interpolated probabilities ===
Chara sizes: 1134840 117980000 3122085300 4112744264 51239712064
=====
5 18 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
=====
5/2 Writing 1000 model files
Name:lmpl Vocab:1082210 KB VRRS:10968 KB RSPMax:1108000 KB user:1855.722 sys:144.19 CPU:799.912 real:306.311
Building binary lm model files
Reading model/corpus.en.1000.bpe.lm.s.arpa
...5...18...11...20...25...30...35...40...45...50...55...60...65...70...75...80...85...90...95...100
=====
OCCURS
Evaluate scores of the training data
***
opt: 6.882
doc: 2.783
jraye@jraye-VirtualBox: ~/Documents/mtnt-master$
```

Evaluated scores of the training data are as follows:

ppl: 6.882

bpe: 2.783

The given below shows the accuracy of the processed and examined data.

```
Activities Terminal
File Edit View Search Terminal Help
jraye@jraye-VirtualBox: ~/Documents/mtnt-master
jraye@jraye-VirtualBox:~/Documents/mtnt-master$ cat NMT/test/test.en-fr.es | python3 analysis/count_keywords.py resources/profanities.es
18
jraye@jraye-VirtualBox:~/Documents/mtnt-master$ cat NMT/test/test.en-fr.es | python3 analysis/count_emails.py
42
jraye@jraye-VirtualBox:~/Documents/mtnt-master$ cat NMT/test/test.en-fr.es | python3 analysis/uk_un_ratio.py
UK
13.7% 44.3%
jraye@jraye-VirtualBox:~/Documents/mtnt-master$
```


For that we need to run it on pure ubuntu terminal environment and we had successfully made and setup it for this project.

Conclusion

We demonstrate a new dataset to test MT models' resistance to the many types of noise found in natural language on the Internet. For two language pairs, English French and English, we submit parallel training and test data in both directions.

For two language pairs,

Japanese, as well as statistics in those languages that are monolingual three distinct languages. We show that this dataset has more noise than existing MT test sets and that it is more difficult to interpret.

Models who have been educated on conventional models have a hurdle to models trained on standard MT corpora is a collection of MT corpora. Furthermore, we show that these

A simple domain adaptation technique will not be enough to solve the obstacles. This is our intention to contribution for the creation of a standard benchmark for resilience in the noise in MT and encourage research on customized models, datasets, and evaluating the metrics for this exact issue or problems.