SILVERDATA SET CREATION USING IBM MODELS

MILTON LIN

Contents

1. Summary of the pdf

- We describe in Section 2 the task and method. This used to produce silver data sets of tagged Swedish `CoNLL-U` files. `sv_lines-ud-dev_model*.conllu`, where `*` could be {`1, 4, 1L, 4L, 14L, 14HMML`}.
- In Section 3, we evaluated our task by considering our performance on alignment scores.
- In Section 5 I documented how I ran GIZA++, in Section 6, I documented how I ran the cloud translation, in Section 4, I described the given **CoNLL-U** dataset. These can be ignored but I hope this would be useful for future reference.

Date: February 13, 2024.

2. Silver data set on part of speech tagging

Traditional methods of creating silver data, follow the methods of Yarowsky et al., [YNW01] and [YN01], [PMH11]. These have the following components

$$\begin{tikzcd} \text{token alignment} \arrow[dr] & \\ & \text{data projection} \\ \text{data labeling} \arrow[ur] & \end{tikzcd} \tag{1}$$

- For token alignment the common methods follow, [Bro+93], [ON03].
- For data labelling, specifically part-of-speech (POS) tagging, is already done for us, which we remark in Section 4.

2.1. Method.

(1) We trained GIZA++ program on different IBM Models with different training schemes.

size/training scheme				
`bitext.*`	**1**	**4**	**14**	**14HMM**
`bitextL.*`	**1L**	**4L**	**14L**	**14HMML**

where the `*` here are `en` or `sv`. `L` stands for additional data from 1032 English lines, translated using Google cloud Section 6.

(2) We then use the bitext files to create alignment files. These are of the form. `onlymodel*` where $* \in \{$`1, 1L, 4,4L, 14L, 14HMML`$\}$. These contribute to token alignemtn in Equation 1.

(3) The data labelling is done for us. So we simply project the data from these label on to the Swedish part using the token alignments. *Note!* The tokenizations from labeled data are not necessarily the same, so this is a potential area of improvement.

3. Evaluation

How are we going to evaluate the `CoNLL-U` projected datasets? We look at the alignment scores. This is because we had *no* gold dataset tagged Swedish data. (This also seems realistic in real life.) There are two main limitations to this: 1. we suppose alignment scores have positive implications for the quality of the projected dataset, and 2. we do not consider the constraints on compute.

Now that we supposed alignment scores is the main form of evaluation, we list a few avenues and discuss the ones we have chosen.

- Matching the tokenization between the projected datasets and the label sets.
- Linguistic modifications to improve alignment, such as morphology.
- More variations of training schemes, we did a basic analysis of this in Section 3.2.
- Larger data sets, we did a basic analysis of this in Section 3.1

3.1. **Effect of size of training dataset.**

- we first discuss the alignment files obtained from only model 1 and only model 4, a Table 1 and discussed the scores. It is interesting to observe that for Model 4 the score is lower.
- We then printed the scores when we enlarge the data, it is slightly consistent that all scores have improved, see Table 2.

Pair Model	**1**	**4**
1	0.247399	0.000765337
2	0.0877891	0.000765405
3	1.87891e-11	1.06507e-25
4	1.32016e-06	2.08414e-18
5	0.000977398	1.17121e-09
6	5.19068e-13	1.4498e-32
7	4.27319e-17	1.30035e-38
8	1.31154e-11	7.12195e-30
9	5.99647e-13	1.33009e-32
10	9.39311e-09	2.4853e-18

Table 1. Alignment Scores comparison between Model 1 and Model 4

It can be seen generally that training only on model 4 has smaller alignment values. Below we also give the alignment scores comparisons between model1L, and model4L.

Pair	Model 1L Score	Model 4L Score
1	0.247292	0.000812804
2	0.100596	0.00102613
3	1.95923e-11	4.90485e-26
4	2.5364e-06	1.62005e-17
5	0.000998972	1.04674e-09
6	5.55109e-13	3.77409e-32
7	1.47609e-16	1.23056e-37
8	2.26524e-11	1.02223e-30
9	6.77975e-13	6.57348e-33
10	1.05341e-08	5.99042e-19

Table 2. Comparison of Alignment Scores for Model 1L and Model 4L

We make the following observation:

- IBM alignment scores from model 1 to model 1L improved. This is expected. And a quick check shows that the quality of new data is not so bad, having both diversity in content and length.
- Model 4 is more complex than model 1, accounting for order and fertility. Additional data might not provide new patterns or information. That could be why scores for Model 4 to Model 4L did not improve generally.

3.2. **Effect on training scheme.** We make the following two basic modifications:

(1) Run model iterations 1+4. Is the combined better than the ones individual?

(2) Run model iterations 1+4+HMM for alignment file. Is running additional HMM better?

We combine all the results into one table.

Pair / Model	1	4	1L	4L	14L	14HMML
1	0.247399	0.000765337	0.247292	0.000812804	0.00532421	0.00730708
2	0.0877891	0.000765405	0.100596	0.00102613	0.00315584	0.0043587
3	1.87891e-11	1.06507e-25	1.95923e-11	4.90485e-26	1.03615e-18	1.86431e-17
4	1.32016e-06	2.08414e-18	2.5364e-06	1.62005e-17	1.65241e-16	2.25778e-12
5	0.000977398	1.17121e-09	0.000998972	1.04674e-09	1.49192e-07	5.40434e-07
6	5.19068e-13	1.4498e-32	5.55109e-13	3.77409e-32	2.24801e-30	9.059e-28
7	4.27319e-17	1.30035e-38	1.47609e-16	1.23056e-37	1.85864e-36	4.66828e-35
8	1.31154e-11	7.12195e-30	2.26524e-11	1.02223e-30	4.16013e-25	3.32206e-24
9	5.99647e-13	1.33009e-32	6.77975e-13	6.57348e-33	1.69728e-30	6.936e-28
10	9.39311e-09	2.4853e-18	1.05341e-08	5.99042e-19	2.72719e-20	4.05154e-18

Table 3. Comprehensive Comparison of Alignment Scores Across Models

We can make a number of observations:

- Adding HMM improves score in general -which can be a quite a scale, see from Model 14L to model 14HMML. In total there is significant improvements from Model 14HMM L.
- When model 4 is added alignment score gets lower. In fact, the best performing is still model 1L.

4. POS dataset

References: Introduction. The dataset consists of English sentence with manually annotated Part-Of-Speech (POS) tags in CoNLL-U format. There are three types of lines.

(1) *Word lines*: containing the annotation of a word/token/node in 10 fields separated by single tab characters. We are interested in the fourth one which is UPOS (Universal part-of-speech tag).

(2) *Blank lines*: these simply mark boundaries.

(3) *comment lines*: `#` gives meta data. In our case, we have the `text` and the `id`.

Let us give a few examples.

```
# newdoc id = 1
# sent_id = en_lines-ud-train-doc1-1
# text = Show All
1 Show show VERB IMP Mood=Imp|VerbForm=Fin 0 root _ _
2 All all PRON TOT-PL Case=Nom 1 obj _ _
```

- Since this is the first line we have `# newdoc`, which means a new document.
- Then we have the word, started with indexing 1[1] and then its lemma.
- "Show" tagged as a verb in imperative mood and
- "All" as a pronoun.

Another example

```
# sent_id = en_lines-ud-train-doc1-2
# text = About ANSI SQL query mode
1 About about ADP _ _ 5 case _ _
2 ANSI ANSI PROPN SG-NOM Number=Sing 5 compound _ _
3 SQL SQL PROPN SG-NOM Number=Sing 2 flat _ _
4 query query NOUN SG-NOM Number=Sing 5 compound _ _
5 mode mode NOUN _ Number=Sing 0 root _ _
```

[1]as is often in python

5. The GIZA++ program

For `MacOS` users like me, this blog post is perhaps the most useful. GIZA++ is an unsupervised word alignment tool - still used in 2010s. Our input is

Definition 5.1. *Parallel texts* are text placed alongside with their translations. These are referred to as *bitexts*. There are various forms.

- Our first input are the following two bitexts, `bitext.en` and `bitext.sv`.
- We use the `enlarge_bitext.py` to add in the developments. To get `bitextL.en` and `bitextL.sv`.

we will use GIZA++ to obtain alignments.

5.2. Compiling GIZA++.

Clone the zipped repository

```
$ git clone git@github.com:moses-smt/giza-pp.git
```

Below are a list of potential modifications you may have to make.

- For case-insensitive file systems like `MacOS` we need to modify `model3.cpp` file.

```
alignfile = Prefix + ".A3." + number ;
  test_alignfile = Prefix + ".tst.AA3." + number ;
```

- You may receive error

```
hmm.cpp:255:6: error: expected expression
    [l](double x){return x*2*l;});
```

  This is an expressed introduced in `C++11`. You may have either go to the relevant file.

```
CFLAGS_OPT = $(CFLAGS) -O3 -funroll-loops -DNDEBUG -
DWORDINDEX_WITH_4_BYTE -std=c++11
```

  or you may delete the line of code if you do not use `hmm`.

5.3. Overview of input files.

Before running GIZA++ we will need to use `./plain2snt.out` to create necessary files. It may not be there as to compile a C++ file like `snt2plain.cpp`, you generally use a C++ compiler such as `g++` (for GNU Compiler Collection) or clang (for LLVM project). In which case use,

```
g++ -o plain2snt.out plain2snt.cpp
```

```
g++ -o snt2plain.out snt2plain.cpp
```

where the former code is what we use to turn `plain` to `snt` files. Now we input our bitext files

```
./plain2snt.out ../en_sv_text_files/bitext.en ../en_sv_text_files/bitext.sv
```

to obtain the following:

- Vocabulary files. `bitext.en.vcb`, `bitext.sv.vcb`.
  Each entry is of the form
  $$(\texttt{uniq\_id1 string1 no\_occurrences1})$$
- Bitext files. Each entry encodes the data of:
  $$(n(p,q), c(p), c(q))$$
  (1) where $n(p,q)$ number of a given sentence pair (p,q) occurred.
  (2) $c(p)$ and $c(q)$, sentence p and qs numerical encoding from `vcb` files.
- Cooccurence file.

```
./snt2cooc.out ../en_sv_text_files/bitext.en.vcb
../en_sv_text_files/bitext.sv.vcb
../en_sv_text_files/bitext.en_bitextL.sv.snt >
../en_sv_text_files/co.en_sv.cooc
```

5.4. **Runing the GIZA++ program.**

-
```
./GIZA++ -S bitext.en.vcb -T bitext.sv.vcb
-C bitext.en_sv.snt -CoocurrenceFile cooc.en_sv
```
- To run only model 1 ,

```
./GIZA++ ../configuration_files/onlymodel1.gizacfg
-model1dumpfrequency 1
-hmmiterations 0
```

  and to run only model 4

```
./GIZA++ ../configuration_files/onlymodel4.gizacfg
-hmmiterations 0
-model345dumpfrequency 1
```

  In the `.gizacfg` files of model 1 and 4 I am able to set which models run. [2] It is also fine additionally setting it within the prompt, which overruns the config file.

Some outputs of the `GIZA++` program. There are many files that get returned. But we mention the ones that are of interest.

- `.gizacfg` file. This configuration file tells GIZA++ how to run: which files to use, how many iterations to perform for each model, where to log its output, and other specific settings.

[2]The `hmm` models doesn't seem to be able to be changed.

•

6. Google Cloud Translation

To set this up:

(1) Create a google cloud project.

(2) Create an API creditial.

- the Python client library for Google Cloud services primarily uses service account credentials for authentication and does not natively support API key authentication in the code
- We will create a service account.
- Create a key following the directions on Google Console.
    - Select `JSON` as the `Key type` and `click Create.`
    - This downloads a service account key file. You cannot download it again.

(3) We now set up the environment. The `export` command is used in Unix-like operating systems' shell environments (like Bash in Linux and macOS) to set environment variables.

```
export GOOGLE_APPLICATION_CREDENTIALS= "/path/file.json"
```

`GOOGLE_APPLICATION_CREDENTIALS` environment variable for the duration of the terminal session. This environment variable tells the Google client libraries where to find the credentials file necessary for authentication.

(4) Anaconda environments can sometimes cause conflicts or issues with certain packages. Consider creating a new, clean environment and installing your dependencies there:

```
conda create --name new_env python=3.10
conda activate new_env
pip install google-cloud-translate grpcio
```

From the above procedures, we have

- Code in `english_swedish_translation.py`
- successfully translated 1032 lines of English text into Swedish, `dev_pos_bitext.sv`.

We will now combine this with the original bitexts to create larger bitext.

References

[Bro+93] Brown, Peter F et al. "The mathematics of statistical machine translation: Parameter estimation". In: (1993) (cit. on p. 2).

[ON03] Och, Franz Josef and Ney, Hermann. "A systematic comparison of various statistical alignment models". In: *Computational linguistics* 29.1 (2003), pp. 19–51 (cit. on p. 2).

[PMH11] Plas, Lonneke van der, Merlo, Paola, and Henderson, James. "Scaling up Automatic Cross-Lingual Semantic Role Annotation". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Ed. by Dekang Lin, Yuji Matsumoto, and Rada Mihalcea. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 299–304. URL: https://aclanthology.org/P11-2052 (cit. on p. 2).

[YN01] Yarowsky, David and Ngai, Grace. "Inducing Multilingual POS Taggers and NP Bracketers via Robust Projection Across Aligned Corpora". In: *Second Meeting of the North American Chapter of the Association for Computational Linguistics*. 2001. URL: https://aclanthology.org/N01-1026 (cit. on p. 2).

[YNW01] Yarowsky, David, Ngai, Grace, and Wicentowski, Richard. "Inducing Multilingual Text Analysis Tools via Robust Projection across Aligned Corpora". In: *Proceedings of the First International Conference on Human Language Technology Research*. 2001. URL: https://aclanthology.org/H01-1035 (cit. on p. 2).