

Project Report

Conversion of Raw Sentence to Grammatically Correct Sentence

Mentor : - Dr Subhash Chandra Pandey

**Aditi (BE/15086/17), CSE
Salvi Verma (BE/15048/17), CSE
7th sem CSE**

1. OBJECTIVE:

Automate the conversion of news headlines into its canonical form (grammatically correct sentences).

Ex: Raw headlines: KSERC to evaluate power tariff details

Grammatically transformed: KSERC is going to evaluate power tariff details

2. INTRODUCTION:

Parallel corpus was prepared by collecting news headlines (raw data) and manually converting them into grammatically correct sentences (transformed data) using a given set of rules. Manual conversion included reordering POS, changing tense forms of verbs, introducing forms of verb 'be', etc. The original raw and transformed corpus is present in the "un-processed" directory.

Using rule-based methods, SMT or a hybrid approach; this process has to be automated. Such formulations can have application in e-reader services, as an accessibility service, etc..

3. CHARACTERISTICS OF NEWS HEADLINE: [*Reference*](#)

News headlines are different grammatically than normal sentences

1. Use present simple tense for past events

The present tense is quick and current, and helps emphasise the action happening, rather than its completion.

- Parliament confirms new stray dog policy
- Lion escapes zoo

2. Leave out auxiliary verbs

With perfect, continuous and passive structures, auxiliary verbs are not necessary. This makes some headlines appear to be in the past tense, when actually the headlines use past participles, or particles, not the past simple.

- Four stranded in sudden flood (four people have been stranded / were stranded)
- Temperatures rising as climate changes (temperatures are rising)

3. Use infinitives for future events

Using the infinitive, a future time is not always necessary to demonstrate the future tense in headlines.

- Parliament to decide new policy tomorrow
- President to visit France for further talks

4. Leave out articles (a, an, the)

- Prime Minister hikes Alps for charity (The Prime Minister hiked the Alps)
- Man releases rabid dog in park (A man released a rabid dog in a park)

5. Leave out "to be"

- Residents unhappy about new road (residents are unhappy)
- Family of murder victim satisfied with court decision (family of murder victim is satisfied.)

6. Leave out "to say"

- Mr Jones: "They're not taking my house!"

7. Replace conjunctions with punctuation

- Police arrest serial killer – close case on abductions
- Fire in bakery: hundreds dead

8. Use figures for numbers

- 9 dead in glue catastrophe

- 7 days to Christmas – shoppers go mad

4. CORPUS PREPARATION

Corpus is present in the “un-processed” directory is prepared as follows:

1. Aligned mismatched raw headlines & their corresponding transformation
2. Some data with only raw, or transformed headlines were pruned
3. Certain nouns and their abbreviations were converted to a common word in the corpus to reduce data sparsity (Chief Minister – CM; Govt – Government)
4. Due to ambiguity in the original corpus, punctuation was removed from sentences to have uniformity

The prepared corpus is present in the “prepared/corpus.tsv” file.

5. SMT

1. Statistical machine translation (SMT) is a machine translation paradigm where translations are generated on the basis of statistical models whose parameters are derived from the analysis of parallel corpora.
2. Moses is an implementation of the statistical (or data-driven) approach to machine translation (MT). This is the dominant approach in the field at the moment, and is employed by the online translation systems deployed by the likes of Google and Microsoft. In statistical machine translation (SMT), translation systems are trained on large quantities of parallel data (from which the systems learn how to translate small segments), as well as even larger quantities of monolingual data (from which the systems learn what the target language should look like).
3. The training process in Moses takes in the parallel data and uses co-occurrences of words and segments (known as phrases) to infer translation correspondences between the two languages of interest. In phrase-based machine translation, these correspondences are simply between continuous sequences of words, whereas in hierarchical

phrase-based machine translation or syntax-based translation, more structure is added to the correspondences.

6. Phrase Based SMT

1. In phrase-based machine translation, Moses takes in the parallel data and uses occurrences of words and segments (known as phrases) to infer translation correspondences between continuous sequences of words.
2. The phrase-based approach has the ability to directly describe relation between strings of different lengths and improves modeling considerably.
3. This provides a direct relation between the source context and the target word selection. In addition, the phrases provide a much better model for local reordering than the distortion probability.
4. In translation, some source words seem to be deleted in translation, and some target words seem to appear out of nowhere. Deletions and insertions are dependent lexically on their context.

In the phrase based models many of the aspects of translation are modeled implicitly.

1. **Phrase translation model:** Given the availability of phrase correspondence (\hat{t}, \hat{s}) , its probability is calculated as maximum likelihood estimation.
2. **Distortion Model:** Suppose, a_i is the starting word position of the source phrase that is translated into the i th phrase position of the target sentence, and b_{i-1} is the ending word position of the source phrase that is translated into the $i - 1$ th position. The below function states the distortion model of PB-SMT.
3. **Lexical weighting model:** A phrase pair can be evaluated based on the probability of the alignments of the internal words. First, a word translation probability distribution is calculated from a word alignment of the parallel corpus using maximum likelihood estimation.
4. **Word penalty:** This model keeps into account the addition of the target word. This helps in language pairs where sentences in one language are shorter/longer.
5. **Phrase penalty:** This model either rewards or punishes each time a phrase pair is used. For example, this model helps in promoting the use of longer

phrases by keeping into account the used phrase pair.

```

aditi@DESKTOP-9EINJRP: ~/working/mert-work
line 0: Translation took 0.001 seconds total
AC
aditi@DESKTOP-9EINJRP: ~/working/mert-work$ ~/moses/bin/moses -f ~/working/mert-work/moses.ini
Defined parameters (per moses.ini or switch):
  config: /home/aditi/working/mert-work/moses.ini
  distortion-limit: 6
  feature: UnknownWordPenalty WordPenalty PhrasePenalty PhraseDictionaryMemory name=TranslationModel0 num-features=4 path=/home/aditi/working/train/model/phrase-
table.gz input-factor=0 output-factor=0 LexicalReordering name=LexicalReordering0 num-features=6 type=wbe-msd-bidirectional-fe-allff input-factor=0 output-factor=0 path
/home/aditi/working/train/model/reordering-table.wbe-msd-bidirectional-fe.gz Distortion KENLM name=LM0 factor=0 path=/home/aditi/lm/train.blm.tr order=3
  input-factors: 0
  mapping: 0 T 0

weight: LexicalReordering0= 0.155112 0.0659919 0.0574687 0.152017 0.160123 0.160086 Distortion0= 0.0590615 LM0= 0.0252456 WordPenalty0= 0.0279785 PhrasePenalty
= 0.0532572 TranslationModel0= 0.00669017 0.0313174 0.0141061 -0.0315451 UnknownWordPenalty0= 1
UnknownWordPenalty0
FeatureFunction: UnknownWordPenalty0 start: 0 end: 0
line-WordPenalty
FeatureFunction: WordPenalty0 start: 1 end: 1
line-PhrasePenalty
FeatureFunction: PhrasePenalty0 start: 2 end: 2
line-PhraseDictionaryMemory name=TranslationModel0 num-features=4 path=/home/aditi/working/train/model/phrase-table.gz input-factor=0 output-factor=0
FeatureFunction: TranslationModel0 start: 3 end: 6
line-LexicalReordering name=LexicalReordering0 num-features=6 type=wbe-msd-bidirectional-fe-allff input-factor=0 output-factor=0 path=/home/aditi/working/train/model/r
eordering-table.wbe-msd-bidirectional-fe.gz
Initializing Lexical Reordering Feature..
FeatureFunction: LexicalReordering0 start: 7 end: 12
line-Distortion
FeatureFunction: Distortion0 start: 13 end: 13
line-KENLM name=LM0 factor=0 path=/home/aditi/lm/train.blm.tr order=3
FeatureFunction: LM0 start: 14 end: 14
Loading UnknownWordPenalty0
Loading WordPenalty0
Loading PhrasePenalty0
Loading LexicalReordering0
Loading table into memory...done.
Loading Distortion0
Loading LM0
Loading TranslationModel0
Start loading text phrase table. Moses format : [0.549] seconds
Reading /home/aditi/working/train/model/phrase-table.gz
---5---10---15---20---25---30---35---40---45---50---55---60---65---70---75---80---85---90---95---100
*****
Created input-output object : [1.625] seconds

```

7. Mathematical Formulation

1. Instead of employing the Bayesian Theorem, the posterior probability $P(t_1^I | s_1^J)$ is expressed as a set of M submodels $h_m(t_1^I | s_1^J)$, which are weighted by a model parameter λ_m .

$$P(t_1^I | s_1^J) = \frac{\exp(\sum_{m=1}^M \lambda_m h_m(t_1^I, s_1^J))}{\sum_{t_1^I} \exp(\sum_{m=1}^M \lambda_m h_m(t_1^I, s_1^J))}$$

2. Since the normalizing denominator only depends on the source sentence, it

has no influence on the maximization of the entire probability. This leads to the following equation.

$$\hat{s}_1^I = \operatorname{argmax}_{e_1^I} P(t_1^I | s_1^J) = \operatorname{argmax}_{e_1^I} \sum_{m=1}^M \lambda_m h_m(t_1^I, s_1^J)$$

3. The model is also known as the log-linear model.

8. Workflow

1. Compiling mooses, GIZA++ in "mooses" & "giza-pp" folder in Home directory respectively.

This link contains all the commands that have been used for mooses installation.

https://drive.google.com/file/d/1oXTbNcmBp4hJeb86Brl_05tuE-0aQW6t/view?usp=sharing

2. Tokenizing, truecasing, cleaning and splitting data (randomizing) in 70-15-15 ratio as Training-Validation-Test sets.

3. Data files should be present in "corpus/split" folder in home directory as Train.clean.rw, Train.clean.tr, Validate.clean.rw, Validate.clean.tr, Test.clean.rw, Test.clean.tr

4. Language Model Training, Training Translation System, Tuning and Testing are done.

This link contains all the commands for this purpose.

https://drive.google.com/file/d/1CY19_eHamh-Bpp2N98jB9wt670EQ2Fl6/view?usp=sharing

9. Language Model Training

1. Moses includes the KenLM language model creation program, Implz.
2. The language model (LM) is used to ensure fluent output, so it is built with the target sentence (i.e Translated Sentences in our case).
3. We have built a 3-gram language model.
4. Then we binarised (for faster loading) the *.arpa.en file using KenLM.

10. TRAINING TRANSLATION SYSTEM

To do this, we run word alignment (using GIZA++), phrase extraction and scoring, create lexicalized reordering tables and create a Moses configuration file, all with a single command.

```
aditi@DESKTOP-9EINJRP:~/l$ ~/moses/bin/implz -o 3 --discount_fallback <~/corpus/split/train.clean.tr > train.tok.arpa.tr
=== 1/5 Counting and sorting n-grams ===
Reading /home/aditi/corpus/split/train.clean.tr
----5---10---15---20---25---30---35---40---45---50---55---60---65---70---75---80---85---90---95---100
tcmalloc: large alloc 1189773312 bytes == 0x55966d3cc000 @
tcmalloc: large alloc 3965886464 bytes == 0x5596b4274000 @
*****
Unigram tokens 21756 types 6743
=== 2/5 Calculating and sorting adjusted counts ===
Chain sizes: 1:80916 2:1801995392 3:3378741248
tcmalloc: large alloc 3378741248 bytes == 0x55966d3cc000 @
tcmalloc: large alloc 1802002432 bytes == 0x5597a1164000 @
Statistics:
1 6743 D1=0.689065 D2=1.12073 D3+=1.40226
2 18775 D1=0.872675 D2=1.30512 D3+=1.63361
3 20177 D1=0.937549 D2=1.56762 D3+=1.48477
Memory estimate for binary LM:
type      kB
probing   965 assuming -p 1.5
probing  1102 assuming -r models -p 1.5
trie      475 without quantization
trie      313 assuming -q 8 -b 8 quantization
trie      456 assuming -a 22 array pointer compression
trie      295 assuming -a 22 -q 8 -b 8 array pointer compression and quantization
=== 3/5 Calculating and sorting initial probabilities ===
Chain sizes: 1:80916 2:300400 3:403540
----5---10---15---20---25---30---35---40---45---50---55---60---65---70---75---80---85---90---95---100
#####
=== 4/5 Calculating and writing order-interpolated probabilities ===
Chain sizes: 1:80916 2:300400 3:403540
----5---10---15---20---25---30---35---40---45---50---55---60---65---70---75---80---85---90---95---100
#####
=== 5/5 Writing ARPA model ===
Name:implz      VmPeak:7026264 kB      VmRSS:1187784 kB      RSSMax:1207484 kB      user:0.163273      sys:6.38807      CPU:6.55134
eal:6.59276
```

11. TUNING

1. Tuning requires a small amount of parallel data, separate from the training data. In our case 15% of the dataset has been used for tuning.
2. The end result of tuning is an ini file with trained weights.

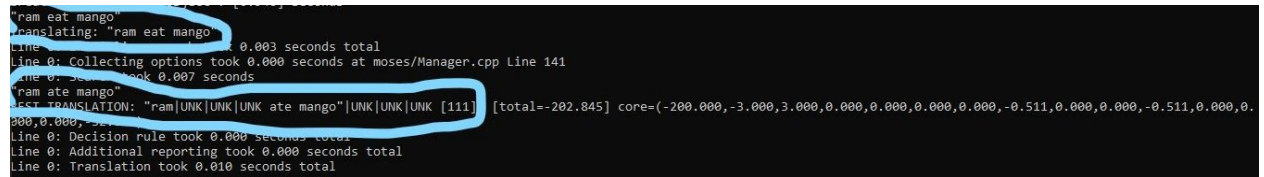
12. TESTING

1. We can now run Moses and type in our raw sentence for a query.

For Eg:

Raw Sentence: ram eat mango.

Translated Sentence: ram ate mango.

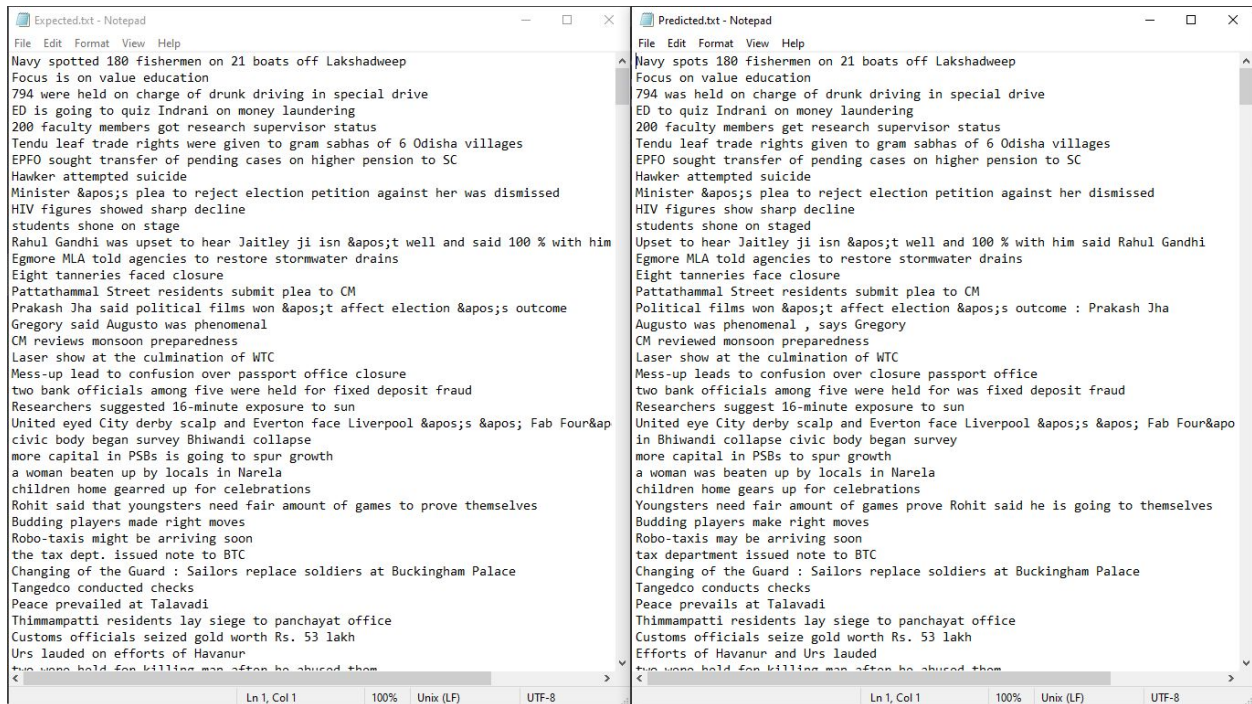
A terminal window showing the execution of the Moses translation tool. The input is "ram eat mango". The output shows the translation process, including the use of the Moses Manager and the resulting translated sentence "ram ate mango". The terminal output is as follows:

```
"ram eat mango"
Translating: "ram eat mango"
Line 0: Collecting options took 0.003 seconds total
Line 0: Collecting options took 0.000 seconds at moses/Manager.cpp Line 141
Line 0: Decision rule took 0.007 seconds
"ram ate mango"
BEST TRANSLATION: "ram|UNK|UNK|UNK ate mango"|UNK|UNK|UNK [111] [total=-202.845] core=(-200.000,-3.000,3.000,0.000,0.000,0.000,0.000,-0.511,0.000,0.000,-0.511,0.000,0.000,0.000,0.000,0.000)
Line 0: Decision rule took 0.000 seconds total
Line 0: Additional reporting took 0.000 seconds total
Line 0: Translation took 0.010 seconds total
```

2. To test the model on our test data which comprises of 15% of our dataset.
3. This creates a test.translated.tr file which can be viewed using any editor.

13. RESULT

1. We compare the files Expected output file and Predicted output file and calculated the BLEU score.



2. The translated sentences obtained were fairly accurate.

3. METRIC

The BLUE Score obtained was 72.60%.

```
[2]+ Exit 2
nohup nice ~/moses/scripts/training/mert-moses.pl ~/corpus/split/validate.clean.rw ~/corpus/split/validate.
~/model/moses.ini --merrtdir ~/moses/bin/ --decoder-flags "-threads all" &> mert.out
aditi@DESKTOP-9EINJRP:~/working$ ~/moses/scripts/generic/multi-bleu.perl -lc ~/corpus/split/test.clean.tr < ~/working/test.translated.tr
BLEU = 72.60, 91.1/78.6/69.4/61.4 (BP=0.977, ratio=0.977, hyp_len=4594, ref_len=4703)
It is not advisable to publish scores from multi-bleu.perl. The scores depend on your tokenizer, which is unlikely to be reproducible fr
oss research groups. Instead you should detokenize then use mteval-v14.pl, which has a standard tokenization. Scores from multi-bleu.pe
al purposes when you have a consistent tokenizer.
```