Omri Zomet 032667693
Ohad Shai 032849390

NLP - project report

Overview

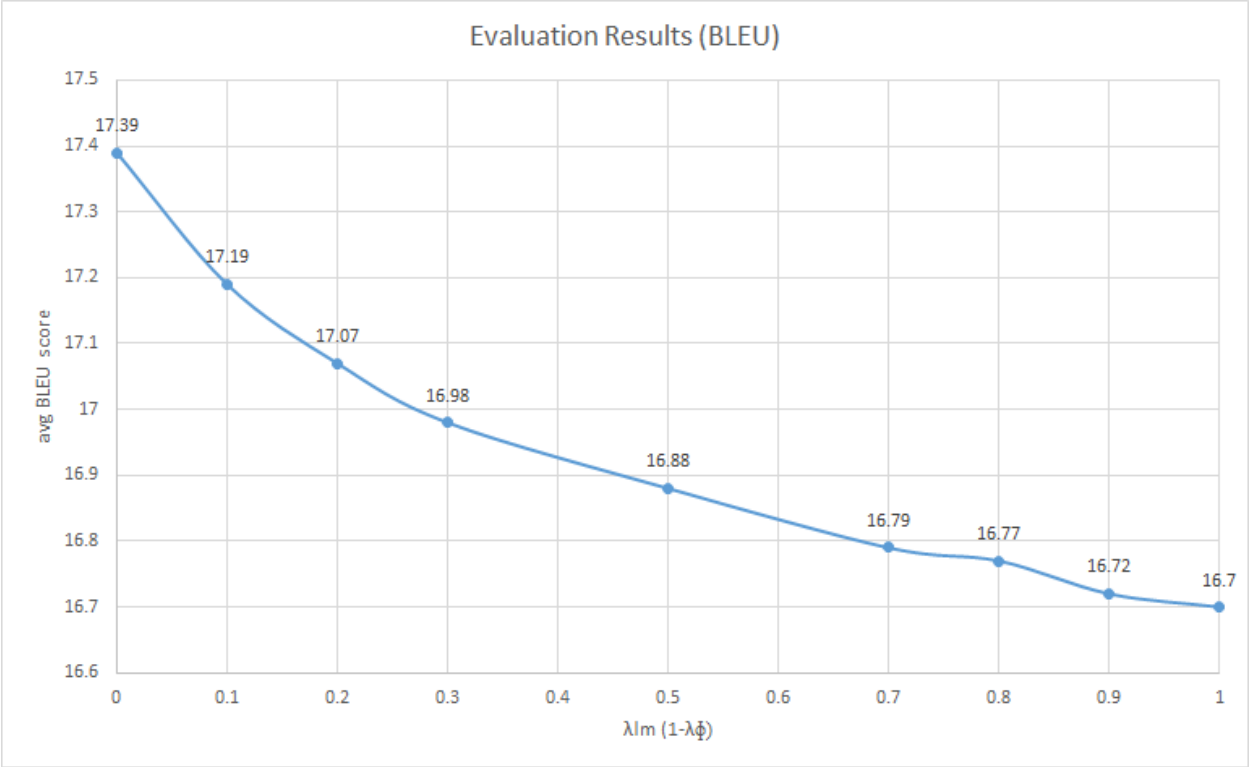
The final project of the course was to develop a fully-automatic translation pipeline, involving some existing off-the-shelf components along with our own algorithms to translate Biblical Hebrew sentences into English.
We implemented the System in Java.

Some general comments regarding our implementation:
- We used Stanford Tokenizer (see link at the end)
- Language model - we have used Witten-Bell smoothing, as it showed better results in Ex1.
- Phrase table - we have used “union of the alignments”.
  There are other ways to implement it (using “cut” of the alignments or some other method) - which might produce better results.
- Stack decoding:
  For unmatched words (words without translation in lattice), we have decided to translate to “NA” word and give 0 probability.
  For pruning, we used histogram based pruning with stack size of 10.
  We didn't do ordering.
  We didn't calculate and use future cost.
  Regarding weighting - we have decided to use the lambdas weighting only on the full coverage hypothesis. Partial hypothesis scores were only concerned with translation score.

Results

The following graph shows the average BLEU score (over 800 sentences in the test set) for different lambda parameters, given stack size = 10 and language model of 3-gram.

Best score was 17.39.



* average BLEU score over 800 sentences in test set

Output from BLEU script:

λ_{Φ}=1, λ_{LM} = 0
BLEU = 17.39, 52.5/24.9/14.7/9.4 (BP=0.845, ratio=0.856, hyp_len=17916, ref_len=20937)

λ_{Φ}=0.9, λ_{LM} = 0.1
BLEU = 17.19, 52.4/24.8/14.5/9.1 (BP=0.843, ratio=0.855, hyp_len=17891, ref_len=20937)

λ_{Φ}=0.8, λ_{LM} = 0.2
BLEU = 17.07, 52.4/24.7/14.4/9.0 (BP=0.844, ratio=0.855, hyp_len=17892, ref_len=20937)

λ_{Φ}=0.7, λ_{LM} = 0.3
BLEU = 16.98, 52.3/24.6/14.3/8.9 (BP=0.844, ratio=0.855, hyp_len=17892, ref_len=20937)

λ_{Φ}=0.5, λ_{LM} = 0.5
BLEU = 16.88, 52.1/24.5/14.1/8.8 (BP=0.845, ratio=0.856, hyp_len=17919, ref_len=20937)

λ_{Φ}=0.3, λ_{LM} = 0.7
BLEU = 16.79, 51.9/24.3/14.0/8.7 (BP=0.848, ratio=0.858, hyp_len=17970, ref_len=20937)

λ_{Φ}=0.2, λ_{LM} = 0.8
BLEU = 16.77, 51.9/24.2/13.9/8.7 (BP=0.849, ratio=0.860, hyp_len=18000, ref_len=20937)
λ_{Φ}=0.1, λ_{LM} = 0.9
BLEU = 16.72, 51.8/24.1/13.8/8.6 (BP=0.851, ratio=0.861, hyp_len=18026, ref_len=20937)

λ_{Φ}=0, λ_{LM} = 1
BLEU = 16.70, 51.5/23.8/13.6/8.5 (BP=0.861, ratio=0.870, hyp_len=18220, ref_len=20937)

* Translation for 800 sentences, with λ_{Φ}=0.9, λ_{LM} = 0.1, can be seen in "translation_lmdTrans_09_lmdLM_01.txt" file.

Analysis

The best results (when not ignoring the language model) were acquired by using λ_{Φ}=0.9, λ_{LM} = 0.1.
This might be explain by 2 major factors:

- The language model is based on non-biblical text - hence it's probably less appropriate for the the development and test data.
- We noticed that the language model log(p) is usually much more "dominant" than the translation log(p) values (much lower negative weights).

As the language model tries to find a better phrasing (as opposed to the translation model that looks for better word translation) - it is reasonable that following the factors mentioned above, a bigger lambda for the translation model will produce a more "accurate" translation (although phrasing might be a bit off).

Perhaps by using 4-gram or 5-gram - the language model can give better results.

In general, as we understood from the forum discussion, a BLEU score between 15 and 30 should be OK for that kind of simple translation system (especially for the test set).

We tried using a language model of 4-gram - however, failed to create one even on our strongest PCs (too much RAM consumption).
We have tried running with different stack sizes and noticed that a bigger stack (e.g. 20 vs 10) can increase the BLEU score very little - BLEU score +0.02 (and not on all runs of different lambdas).

In order to overcome words with no translation (like names - see example below) - we have tried translating the words as-as, hoping that the phonetic similarity will improve the results. However, that didn't change the BLEU score.

Examples
Some of the sentences are translated almost spot on:

Sentence #2:
Reference: “And the heathen shall know that I the LORD do sanctify Israel, when my sanctuary shall be in the midst of them for evermore.”
Translation: “and the heathen shall know that I the LORD do sanctify Israel when my sanctuary in the midst of them for evermore.”

Sentence #367:
Reference: “And there was war between Asa and Baasha king of Israel all their days.”
Translation: “And there was war between Asa and Baasha king of Israel all their days.”

Sentence #712 (almost..):
Reference: “But Abner the son of Ner, captain of Saul's host, took Ish-bosheth the son of Saul, and brought him over to Mahanaim;”
Translation: “and Abner the son of Ner, captain of the host which in hell, took Ish-bosheth the son of Saul, made him pass Mahanaim”
* Saul’s == hell…

Erroneous translations examples:

Missing vocabulary example:
Reference: “And Gera, and Shephuphan, and Huram.”
Translation: “<UNK> and <UNK> and Huram”

Both translation and model weighting fail (0.1/0.9 and vice versa)
Reference: “Thou didst walk through the sea with thine horses, through the heap of great waters.”
Translation preference: “<UNK> in the sea, thine horses and thy ass many waters.”
LM preference: “<UNK> in the sea, thine horses and thy clay waters,”
Both fail to translate beginning of the sentence - and then you either choose “ass waters” or “clay waters”

Inaccurate words, bad linguistic (“the the the the”)
Reference: “For the name of God is blasphemed among the Gentiles through you, as it is written.”
Translation: “for there God for you out them that are slain in the the the the heathen: as it is written,”

Ideas for improvements:
- Using reordering for cost calculations during the stack decoding
- Using synonyms (especially for verbs, as we have seen several verbs that translated to NA)

- Using mapping of names or uncommon phrases (to tackle words without translations).
- Using a more suitable and/or bigger corpus for language model.
- Clean bad linguistic (“the the” doesn’t sound right...)
- Stemming might help matching verbs and nouns.

Resources notes

We had to use stronger PCs (either from work or University labs), as Java required a lot of RAM (and still took very long to run) in order to create the language model, loading it or loading the phrase table.
(Due notice that Java memory flags [ie: -Xmx5g] should be used in case machine is too low on resources)

References and External Libraries

Course Site:
http://www.cs.tau.ac.il/~lenadank/nlp/
Stack decoding presentation:
http://cs.jhu.edu/~post/mt-class/2014/mt-class-2014-02-25-stack-decoding.pdf
The Alignment Template Approach to Statistical Machine Translation:
https://files.ifi.uzh.ch/dbtg/sdbs13/T07.0.pdf
Giza++:
https://code.google.com/p/giza-pp/
Phrase based models:
http://www.statmt.org/book/slides/05-phrase-based-models.pdf
Stack decoding:
http://www.statmt.org/book/slides/06-decoding.pdf
Stanford Tokenizer
http://nlp.stanford.edu/software/tokenizer.shtml
log4j:
http://logging.apache.org/log4j/1.2/
jcommander
http://jcommander.org/
guava
https://code.google.com/p/guava-libraries/