Praktikum Maschinelle Übersetzung: Phrase Extraction and Decoder

Um die Aufgaben auszuführen, können Sie ihre Daten in folgendem Verzeichnis speichern:

/project/smtstud/ss17/systems/USERNAME/

Wir werden die beiden asymmetrischen word alignments, die wir das letzte Mal trainiert haben, mit verschiedenen Kombinationsheuristiken kombinieren und jeweils einen Phrase Table extrahieren.

Für die Kombination der word alignments und zum Extrahieren der phrase tables verwenden wir das weit verbreitete SMT Toolkit **moses**. (http://www.statmt.org/moses/)

Wenn Sie die IBM word alignment models nicht in der letzten Praktikumsstunde trainiert haben, führen Sie das script: /project/smtstud/ss17/bin/copyGizaModels.sh in Ihrem Userverzeichnis aus. Bitte melden Sie sich nach Abschluss jeder Aufgabe (oder ggf. Zwischenaufgabe), sodass wir ausschließen können, dass Sie mit fehlerhaften Dateien weitermachen.

1. Kombinieren der Word Alignment Modelle

Das Kombinieren der beiden word alignment models aus dem GIZA++ Training wird im moses Training im Schritt 3 erledigt: http://www.statmt.org/moses/?n=FactoredTraining.AlignWords

Rufen Sie den Schritt 3 des moses Trainingsscript auf. Fügen Sie zu folgendem Programmaufruf den Parameter **--alignment** hinzu und generieren Sie drei kombinierte Alignments für die drei verschiedenen Kombinationsheuristiken: **grow-diag-final-and**, **intersection** und **union**.

/project/smtstud/ss17/bin/giza/train-factored-phrase-model.PGIZA.Adapt.perl --root-dir . --corpus corpus/train.de-en.1-99 --f de --e en --first-step 3 --last-step 3

Führen Sie den Programmaufruf in Ihrem Verzeichnis aus. Das moses Trainingsscript schreibt die resultierenden Alignments in das Unterverzeichnis model/. Die Dateinamen reflektieren die Kombinationsheuristik.

1.a) In den resultierenden Alignmentdateien (**model/aligned.grow-diag-final-and**, **model/aligned.intersection** und **model/aligned.union**) befindet sich ein Eintrag pro Wort-zu-Wort Link. Um die Gesamtzahl aller gesetzten Links zu erfahren, können Sie **wc -w <file>** verwenden.

- Wie viele Alignmentlinks wurden jeweils gesetzt?
- Wie unterscheiden sich die drei Heuristiken und welchen Einfluss hat dies auf die Anzahl der generierten Links?

Heuristik	# Links
intersection	
union	
grow-diag-final-and	

2. Phrase Extraction and Scoring

Die Phrasenpaare für das Übersetzungsmodell werden im moses Schritt 5 extrahiert und im Schritt 6 gescored. Das für das Scoring benötigte Lexikon wird in Schritt 4 erzeugt.
Schritt 5: http://www.statmt.org/moses/?n=FactoredTraining.ExtractPhrases
Schritt 6: http://www.statmt.org/moses/?n=FactoredTraining.ScorePhrases

2.a) Führen Sie die Schritte 4 bis 6 zunächst für das mit grow-diag-final-and erzeugte Alignment aus: (Laufzeit ca. 10min)

/project/smtstud/ss17/bin/giza/train-factored-phrase-model.PGIZA.Adapt.perl --root-dir . --corpus corpus/train.de-en.1-99 --f de --e en --alignment grow-diag-final-and --first-step 4 --last-step 6

Sehen Sie sich den Inhalt des resultierenden Phrase Table **(zless model/phrase-table.0-0.gz)** einmal an.

- Was steht in den ersten fünf Feldern (zwischen |||) jeder Zeile? http://www.statmt.org/moses/?n=FactoredTraining.ScorePhrases

2.b) Analysieren sie den Phrase Table mit dem script:
zcat model/phrase-table.0-0.gz | python /project/smtstud/ss17/bin/countPPinPT.py

Sehen Sie sich die erzeugte phrase pair Statistik an. Eine solche Statistik ist nützlich, um auf den ersten Blick zu sehen, ob im Training etwas schief gelaufen ist oder um Phrase Tables zu vergleichen.

- Warum sind die höchsten Einträge um die Diagonale der Tabelle?
- Was ist der Unterschied zwischen der ersten und der zweiten Statistik?
- Phrasen welcher Länge kommen in der ersten Statistik am häufigsten vor, welche in der zweiten und wieso?

2.c) Sehen Sie sich die möglichen Übersetzungen für drei Beispiele an:

zcat model/phrase-table.0-0.gz | grep -P '^die Insel \|' | less
zcat model/phrase-table.0-0.gz | grep -P '^Zimmer \|' | less
zcat model/phrase-table.0-0.gz | grep -P '^, \|' | less

- Wie viele verschiedene Übersetzungen gibt es für „die Insel“, „Zimmer“ und das Komma? (Hinweis: **| wc -l**)

- Halten Sie die Anzahl von Übersetzungen für das Komma für sinnvoll? Was kann man dagegen unternehmen?

phrase	# translations
die Insel	
Zimmer	
,	

Benennen Sie den Phrase Table um und löschen Sie erzeugte Zwischendateien:

```
mv model/phrase-table.0-0.gz model/phrase-table.0-0.grow-diag-final-and.gz
rm model/extract.*
rm model/lex.*
rm model/phrase-table.0-0.stat.half.*
```

2.d) Erzeugen Sie nun nach demselben Muster jeweils einen Phrase Table für die beiden verbleibenden Alignmentheurisitken aus Aufgabe 1. (Vergessen Sie nicht, dazwischen die erzeugten Dateien wie oben umzubenennen und zu löschen.)

Vergleichen Sie die drei Phrase Tables:

- Wieviele Einträge haben die drei Phrase Tables?
- Wie verhalten sich diese Anzahlen zu den Anzahlen der Alignment Links aus Aufgabe 1 und weshalb?

Phrase table	# entries
intersection	
union	
grow-diag-final-and	

3. Decoder

Der Decoder wird mittels einer Parameter-Datei konfiguriert. Kopieren Sie die Datei /project/smtstud/ss17/data/param/Initial.Param in Ihr Arbeitsverzeichnis.
Um diese Parameter-Datei benutzen zu können müssen Sie einige Parameter-Einstellungen festlegen:

SentenceList: Hier geben Sie die Datei an, die die Sätze enthält, die übersetzt werden sollen. In unserem Fall ist es die zu Beginn erwähnte Datei smalltest.de (mit vollständigendem Pfad)

In der zu übersetzenden Datei muss jeweils pro Zeile ein Satz stehen.

ReferenceList: Für die Optimierung wird hier die Datei mit den Referenzen angegeben. Im Testfall haben wir keine Referenzen, deshalb geben wir hier einfach noch mal die gleiche Datei an.

LMFile: Hiermit wird das Language Model spezifiziert. Geben Sie hier Ihr bestes Sprachmodell aus dem Language Model Praktikum an. Falls Sie dies nicht mehr haben, verwenden sie folgendes Sprachmodell:
/project/smtstud/ss17/models/train.de-en.1-99.en.4gram.no-cutoff34.srilm

PhraseTableFileName: Hier geben Sie die Phrasentabelle an. Verwenden Sie die Phrasentabelle aus der letzen Übung aus Aufgabe 2.
Falls Sie diese nicht mehr haben, verwenden Sie folgende Phrasetabelle:
/project/smtstud/ss17/models/phrase-table.0-0.grow-diag-final-and.pruned.sttk

Damit haben Sie ihre Parameter-Datei fertig gestellt. Erzeugen Sie nun die Übersetzungen mittels folgendem Programmaufruf und schauen Sie sich die Log-Datei an:
/project/smtstud/ss17/bin/TranslateMM.static -f Initial.Param > LogFile

Die Datei ist folgendermaßen aufgebaut:
Zunächst werden die Parametereinstellungen, die benutzt werden, angegeben.
Danach werden die Modelle geladen und der Decoder initialisiert.
Danach wird mit der Übersetzung begonnen.
Aufgabe 3.a)
Auf welchen Wert wurde das Reordering Window gesetzt?

Aufgabe 3.b)
Wie viele Einträge enthält die Phrasentabelle? Wie viele verschiedene Quell- und Zielphrasen tauchen in der Phrasentabelle auf?

Aufgabe 3c)
In der Zeile „Prepare Scaling Factors“ sind die Gewichte für die einzelnen Modelle des log-linearen Modells angegeben. Kurz darauf werden die Anfangsindizes der Gewichte für die einzelnen Modells angegeben (Zeile Start LMs und folgende). Dabei stehen die Abkürzungen für Language Model, Distortion Model, Wordcount Model, PhraseCount Model, Sentence Length Model und Translation Model.

Welche Gewichte werden für die einzelnen Modelle benutzt?

Wieso ist das Gewicht für das Wordcount Modell negativ?

Wieso gibt es 4 Gewichte für das Translation Model?

Aufgabe 3.d)
Schauen Sie sich die Übersetzung für den letzten Satz genauer an. Wie wurde dieser Satz aus den vorhandenen Phrasenpaaren aufgebaut? Welcher Score wurde dieser Übersetzung zugewiesen?

4. Beam Size (Zusatzaufgabe):

In dieser Aufgabe möchten wir die Auswirkungen der Größe des Beams auf die Laufzeit des Decoders betrachten. Dazu benutzen wir den Parameter BeamSize. Verwenden Sie die Parameter-Datei aus Aufgabe 1 und übersetzen Sie das Testset mit den Beamsizewerten 1, 10, 100 und vergleichen Sie die benötigte Zeit.

5. Optimierung (Zusatzaufgabe)

Wie in der Vorlesung erwähnt können wir die Gewichte für das log-linear Modell auf einem Development Set optimieren. Dazu benutzen wir folgendes Development Set:
/project/smtstud/ss17/data/dev/dev.btec.de.Final.sc
/project/smtstud/ss17/data/dev/dev.btec.en.Final.sc
Die deutsche Version wird mit dem Parameter SentenceList angegeben und die englische Version mit dem Parameter ReferenceList.
Als Initiale Gewichte benutzen wir 1.0_0.5_-1.0_1.0_0.1_0.1_0.1_0.1 und aktivieren die Optimierung indem der Parameter MEROptimize auf 1 gesetzten wird.

Da die Optimierung länger dauern würde, können Sie sich direkt den entsprechenden Log-file unter folgendem Link anschauen:
/project/smtstud/ss17/data/dev/Optimize.Log

Aufgabe 5a)
Am Ende der Log-Datei sind die Ergebnisse zusammengefasst. Jede Zeile in der Tabelle entspricht einer Iteration. Die zweite Zahl gibt den BLEU Score an. Welcher BLEU-Score wurde in der ersten und letzten Iteration erreicht?

Aufgabe 5b)
Schauen Sie sich die Übersetzungen in den verschiedenen Iterationen für den 25. und 34. Satz an. Das können Sie am einfachsten mittels des Befehls:
grep "TRANSLATION_RESULT_HYP *25 1" Optimize.Log
grep "TRANSLATION_RESULT_HYP *34 1" Optimize.Log