Sandra Sánchez Páez
CLT Modulprojekt
Matrikelnummer 807996
07.04.2022

Zunächst muss ich sagen, dass ich in den Wochen, in denen ich an diesem Projekt gearbeitet habe, sehr viel gelernt habe. Ich konnte die im Kurs erarbeiteten Konzepte in die Praxis umsetzen, und ich habe weitere wichtige Konzepte gelernt, bearbeitet und verinnerlicht. Die wichtigste dieser Lektionen ist jedoch die Notwendigkeit, klar strukturiert zu arbeiten, das Skript durch eine Hauptfunktion auszuführen und von weniger zu mehr zu schreiten, um jeden Schritt zu gewährleisten.

In diesem Sinne sind die große Entdeckung, die ich gemacht habe, die Unit-Tests, mit denen ich relativ spät begonnen habe und die mir geholfen haben, Fortschritte zu machen, ohne Teile des Skripts, die bereits funktionierten, zu verderben. Am Ende habe ich eine Zeit lang aufgehört, sie zu benutzen, und habe gemerkt, dass das ein Fehler war, denn sie hätten mir geholfen, Fehler leichter zu erkennen. Ich sehe auch, wie Tests bei der Untersuchung der Leistung der einzelnen Funktionen helfen können. Die Aufteilung jeder mehr oder weniger komplexen Funktion in mehrere vereinfachte Funktionen ermöglicht es, an verschiedenen Optimierungsmöglichkeiten zu arbeiten.

Die größten Herausforderungen bei diesem Projekt waren die Effizienz und das Verständnis des Hauptalgorithmus. Ich gebe zu, dass ich viele Stunden damit verbracht habe, ihn gut zu verstehen, auch in dem Bestreben, einen effizienten Algorithmus zu definieren, aber als ich merkte, dass mir die Zeit davonlief, beschloss ich, den Pseudocode wörtlich zu befolgen, denn obwohl Effizienz sehr wichtig war, hielt ich es für das Wichtigste, das Programm zum Laufen zu bringen.

Die unmittelbare Folge dieser Entscheidung, die mich viele Tage gekostet hat, ist, dass das Programm sehr lange braucht, um eine so große Menge an Daten zu verarbeiten, dass es nicht möglich ist, Ergebnisse zu erhalten, mit denen man in der zweiten Phase des Projekts arbeiten kann. Ich beschloss schließlich, mit einer angemessenen Datenmenge für den Algorithmus zu arbeiten, mit dem ich arbeitete.

Das Hauptproblem des IBM-Modells 1, das Wort-für-Wort-Übersetzungen erzeugt, besteht darin, die möglichen Übersetzungen von Wörtern herauszufinden, da es weder über die Alignments noch über die Übersetzungen verfügt. Die Lösung besteht darin, die Wortkookkurrenzen zu zählen und die Wahrscheinlichkeiten bei jeder Iteration zu maximieren, bis die Daten konvergieren.

Ein weiterer Rückschlag, der mich gegen Ende des Projekts beschäftigte und der durch eine Inspektion aller Daten vor Beginn der Arbeit hätte vermieden werden können, bestand darin, dass ich entdeckte, dass die Sätze des Goldstandards Satzzeichen enthalten, so dass ich fast alles, was ich bis dahin in der Tokenisierungsphase gemacht hatte, verwerfen musste. Bis dahin hatte ich nltk's wordtokenize verwendet, weil es effizienter ist und weniger Platz benötigt. Mit diesem Tool konnte ich den gesamten Korpus innerhalb von 5 Minuten tokenisieren.

Zuerst habe ich versucht, mit dem parallelen Korpus zu arbeiten, indem ich die Satzpaare in Listen gruppierte, aber das funktionierte wegen Speicherproblemen nicht, so dass ich schließlich mit Tupeln arbeitete.

Um das Vokabular für jede Sprache zu erstellen, habe ich verschiedene Dinge ausprobiert. Itertools erwies sich nicht als effizienter, und am Ende habe ich diese Funktion mit set().union vereinfacht, was mir einige for-Schleifen erspart hat.

Das Listenverständnis hat mir auch geholfen, an der Effizienz zu arbeiten, obwohl ich wegen der Komplexität der Operationen und um die Lesbarkeit zu verbessern, oft auf for-Schleifen zurückgreifen musste. Apropos "for"-Schleifen: Ich habe die Verwendung von "while" vermieden, weil sie ebenfalls zu langsam ist. Ich habe keine Vorteile in der Verständlichkeit des Wörterbuchs gesehen, daher habe ich es der Klarheit halber vermieden.

Wie ich bereits angedeutet habe, stellt der zentrale Algorithmus (Erwartungsmaximierung) das größte Effizienzproblem dar, da er so viele Permutationen erzeugen muss. Ich habe jedoch keine effizientere Datenstruktur als Wörterbücher finden können. Ich habe auch versucht, mit Arrays zu arbeiten, aber der Prozess wurde wegen Speichermangels automatisch unterbrochen.

Ich sehe an mehreren Stellen im Algorithmus Optimierungspotenzial, das ich nicht rechtzeitig richtig kodieren konnte. Einerseits wissen wir, dass zu Beginn jedes Wort in der Zielsprache mit gleicher Wahrscheinlichkeit eine Übersetzung eines Wortes in der Ausgangssprache ist. In Anbetracht dessen wäre es jedoch nicht notwendig, sie zu kodieren und einfach die Wahrscheinlichkeiten a posteriori zu aktualisieren. Dies würde die erste Iteration beschleunigen. Die Wahrscheinlichkeiten könnten auch dank der Deepcopy-Funktion zugänglich sein.

Andererseits werden sowohl s_total als auch total (beide Wörterbücher) für jeden Wert einheitlich initialisiert, so dass wir wie oben vorgehen und nur die Wahrscheinlichkeiten aktualisieren könnten.

Ein weiterer Optimierungsversuch bestand darin, die vorverarbeiteten Korpora und Vokabulare zu speichern, aber mir wurde klar, dass dieser Schritt bei einem kleinen Korpus den Prozess unnötig verlangsamen würde, anstatt ihn zu beschleunigen. Deshalb habe ich mich schließlich entschlossen, alles im selben ersten Schritt zu verarbeiten ('learn_alignments').

Was die Tests anbelangt, so habe ich begonnen, für jede kleine Funktion Tests durchzuführen, vor allem, um den em-Algorithmus zu testen und zu optimieren, was nicht funktioniert hat. Schließlich gibt es eine Datei mit Tests von Hilfsfunktionen und andere mit Tests, die sich auf jede der drei Phasen des Hauptprogramms beziehen.

In der Endphase stellte ich, wie bereits erwähnt, fest, dass die Referenzsätze für die Bewertung die Punktzahl beibehielten, so dass ich meine ausgefeilteren Funktionen zur Vorverarbeitung der Token verwerfen musste. Mir ist auch aufgefallen, dass das Format der Daten in der Datei mit den goldenen Alignments verwirrend war, und ich habe den Code angepasst (sowohl die goldenen Alignments als auch die Alignments, die mein Algorithmus berechnet), so dass ich die Daten kaufen konnte. Der Goldstandard sieht das Format "quellsprache zielsprache" vor, und ich habe mit "zielsprache quellsprache" gearbeitet. Ich fand es praktisch, die Positionen zur besseren Lesbarkeit umzukehren.

Die nächste Herausforderung bestand darin, die Zuordnungen zu verstehen, da ich keine Zuordnungen mit 0 fand, sondern in vielen Fällen mehrere mögliche Zuordnungen für dasselbe zielsprachliche Wort. Ich vermute, dass dies der Grund dafür ist, dass meine Werte für Recall und Precision so niedrig sind. Um die beiden Datensätze kompatibel zu machen, habe ich eine Funktion erstellt, die die Zuordnungen vervollständigt, wenn 0 nicht vorhanden ist.

Die Daten, die ich zum Testen der Funktionen während des gesamten Projekts verwendet habe, waren 3 Sätze mit einer englischen Ausgangssprache und einer französischen Zielsprache. Der Parallelkorpus ist im Repository enthalten. Das reale Modell wurde mit dem parallelen Englisch-Spanisch-Korpus des Europarl-Korpus trainiert, und die Evaluierung wurde mit dem manuell annotierten Englisch-Spanisch-Korpus[1] durchgeführt.

Um ein wenig über die Daten und die Ergebnisse zu sprechen, kann ich sagen, dass ich zuerst versucht habe, mit dem gesamten Korpus zu arbeiten, dann mit der Hälfte davon, und dann bin ich auf 10, 0,5, 0,1 und noch kleinere Anteile heruntergegangen, da der Algorithmus zu lange brauchte, um Ergebnisse zu liefern.

Als ich versuchte, einen Korpus von 1000 statt 10000 Sätzen auszuwählen, erhielt ich Ergebnisse in weniger als zwei Minuten. Nach 3 Iterationen sehe ich, dass die meisten der Wortkombinationen eine Wahrscheinlichkeit von 0,0 haben.

---

[1] [1]Englisch-Spanisch: https://www.hlt.inesc-id.pt/w/Word_Alignments

In der Evaluierung überprüfen wir die Genauigkeit der berechneten Daten, indem wir sie mit einem parallelen Korpus mit manuellen Annotationen vergleichen (die auch hinzufügen, ob jedes Alignment "sicher" oder "möglich" ist).

Bei der Arbeit mit Englisch als Ausgangssprache und Spanisch als Zielsprache ergeben sich die folgenden Ergebnisse

Erstens, mit einem Korpus von 1000 Sätzen und 3 Iterationen:

Recall: 0,07903402854006586
Precision: 0,11934900542495479
AER: 0.8988596996926127913

Recall ist der Anteil der als "sicher" eingestuften Alignments, die vom Modell korrekt berechnet wurden. Die Präzision hingegen ist der Anteil der möglichen Übereinstimmungen an allen Übereinstimmungen. Wenn wir davon ausgehen, dass sowohl "mögliche" als auch "sichere" Ausrichtungen möglich sind, sollte diese Zahl höher sein als die vorherige, was wir auch gefunden haben. Ein ungefährer Wert von 0,2 (0 bedeutet, dass es keine Ähnlichkeit gibt; 1 bedeutet, dass es identisch ist) bedeutet jedoch, dass unsere Ergebnisse nicht gut sind.

Die AER ist eine Kombination aus Recall und Precision, wobei 1 für eine perfekte Ausrichtung steht. Ein Wert von 0,89 würde daher auf eine recht gute Ausrichtung hindeuten.

Der Versuch, mit demselben Korpus zu arbeiten, aber die Iterationen zu erhöhen, führt zu den folgenden Ergebnissen:

P: 0.07135016465422613
R: 0.10940325497287523
A: 0.9077838373822509

Der Anstieg von einem AER von 0,89 auf einen AER von 0,90 bestätigt, dass die Daten umso korrekter sind, je höher die Anzahl der Iterationen ist.

Wenn ich mit einem Korpus von 3000 Sätzen arbeite, stelle ich fest, dass sich der Zeitaufwand exponentiell vervielfacht (8 Minuten für die erste Phase im Vergleich zu zwei Minuten für den Korpus von 1000 Sätzen).

Die Werte sind nur geringfügig günstiger als bei der Arbeit mit 1000 Sätzen, aber ähnlich wie bei 1000 Sätzen und 5 Iterationen.

R: 0.07464324917672886
P: 0.12025316455696203
AER: 0.9003470500743679

Betrachtet man die Übersetzungswahrscheinlichkeiten von zwei wirklich parallelen Wörtern, wie z. B. "nützlich" und "nützlich", denen eine Wahrscheinlichkeit von 0,0568666636950274221 zugewiesen wurde, sieht man jedoch, dass die Ergebnisse nicht sehr zuverlässig sein können.

Hier ist der Link zu meinem Repository (die letzte Übergabe, die erscheint, ist vom 6. April, es gibt 7 in der Warteschlange, deshalb hänge ich auch einen Link zu meinem Dropbox-Verzeichnis an: **https://www.dropbox.com/s/sr82emtxk3by6u4/clt21_sandra_sanchez.zip?dl=0**):

https://gitup.uni-potsdam.de/sanchezpaez/clt21_sandra_sanchez/-/tree/master