Programmierprüfung SS20 (Teil I)

20. April 2020

Nach der Bearbeitung geben Sie Ihren **dokumentierten** Code bis spätestens 13:59 bei Moodle ab.
Wenn Sie diesen Prüfungsteil bestehen, werden Sie zur mündlichen Prüfung eingeladen.

Aufgabe

Auf Moodle neben der Prüfungsaufgabe finden Sie zwei "parallele" Korpora:

- `out.txt`: der Output eines Machine-Translation Systems (MT)
- `ref.txt` die Übersetzungen eines Menschen

Wir wollen sehen, wie gut das MT System ist anhand eines Vergleichs von `out.txt` und `ref.txt`. Dafür werden wir den BLEU Score nehmen [Papineni et al., 2002].

Am Ende sollte das Programm so abrufbar sein (zum Beispiel mit python):

```
python bleu_score.py pathto/can.txt pathto/ref.txt
```

Dabei sollte **ein Score pro Satzpaar ausgegeben werden**.
WICHTIG: Für die Lösung der Aufgabe dürfen Sie keine Funktion aus Paketen / Libraries benutzen, die Ihnen "out-of-the-box" den BLEU Score ausrechnet! Um auf der sicheren Seite zu sein und somit herausfordernde Fragen bei dem Interview zu vermeiden, befolgen Sie am liebsten die Anleitung weiter unten.

Abgabe bei Moodle

Dokumentierter Source Code und der BLEU Score der ersten fünf Satzpaare.

Der BLEU score, schrittweise Anleitung mit Beispielen

Für unseren BLEU score müssen wir uni- bi- und tri-gramme aus zwei Sätzen extrahieren, diese miteinander vergleichen und die "matches" zählen und in eine Formel einfügen. Das ist im Prinzip alles.
Es folgt eine schrittweise grobe Anleitung (an die man sich nicht exakt halten muss).

1. Machen Sie sich mit dem Format der Korpora vertraut. Die `.txt` Dateien sind nicht tokenisiert. Tokenisieren Sie diese mit dem Paket / der Library Ihrer Wahl oder schreiben Sie den Code dafür von Scratch.
2. Gegeben seien zwei Beispiele von Listen:

```
reference = [''Mein'', ''Haus'', ''ist'', ''in'', ''der'', ''Heidelberger'', ''Altstadt'']
```

`candidate = [‘‘Ein’’, ‘‘Haus’’, ‘‘ist’’, ‘‘der’’, ‘‘Heidelberg’’, ‘‘Altstadt’’]`

Schreiben Sie eine Funktion die die n-gramme bis $n = 3$ aus einer solchen Liste extrahiert. Dabei dürfen Sie **keine Pakete** benutzen, die Ihnen direkt die n-gramme extrahieren! Schreiben Sie deshalb selbst den Code, der aus einer Liste / einem Array eine neue Datenstruktur mit den n-grammen erzeugt.

INPUT: eine liste mit tokens, eine zahl n;

OUTPUT: eine liste mit n-grammen

1-gramme, reference: `[‘‘Mein’’, ‘‘Haus’’, ‘‘ist’’, ‘‘in’’, ‘‘der’’, ‘‘Heidelberger’’, ‘‘Altstadt’’]`

2-gramme, reference: `[‘‘Mein Haus’’, ‘‘Haus ist’’, ‘‘ist in’’, ‘‘in der’’...]`

3-gramme, reference: `[‘‘Mein Haus ist’’, ‘‘Haus ist in’’, ‘‘ist in der’’,....]`

3. Nun brauchen wir eine Funktion um den Brevity Penalty BP zu berechnen

   INPUT: zwei Listen;

   OUTPUT: eine Zahl zwischen 0 und 1

   Wenn das system eine kürzere Übersetzung geliefert hat als gewünscht, gibt es eine Bestrafung:

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{1}$$

   wobei in diesem Fall $c = 6$ für die Länge von `candidate` ist und $r = 7$ für die Länge der `reference` ist.

4. Nun brauchen wir eine Funktion die den Anteil an korrekten n-grammen berechnet gegeben `candidate` und `reference`.

   INPUT: zwei Listen mit n-grammen;

   OUTPUT: eine Zahl zwischen 0 und 1

   p_n = |Schnittmenge aus n-grammen in `candidate` und n-grammen in `reference`| / |n-gramme in `candidate`|

$$p_n = \frac{|n\text{grams}(\text{candidate}) \cap n\text{grams}(\text{reference})|}{|n\text{grams}(\text{candidate})|} \tag{2}$$

5. Schließlich können wir BLEU zwischen `candidate` und `reference` berechnen:

   INPUT: zwei Listen mit tokens;

   OUTPUT: eine Zahl zwischen 0 und 1

   Mit $w_n = 0.33\ \forall n$ und $N = 3$:

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{3}$$

6. Nun kann man über die tokenisierten Satzpaare iterieren und für jedes Paar den BLEU score berechnen.

   Beachten Sie dabei dass es zu einem Fehler kommen kann z.B. weil $\log(0) = -\infty$. Finden Sie dafür irgendeine Lösung.

Literatur

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.