Lehrstuhl für Informatik 6
der RWTH Aachen
Prof. Dr.-Ing. H. Ney

Aachen, den 20. Mai 2010
M. Huck, D. Stein

Software-Projektpraktikum Maschinelle Übersetzung 3. Übung

Thema:

Da die menschliche Bewertung von übersetzten Sätzen aufwendig und teuer ist, wird sie nur selten durchgeführt, und, wo es möglich ist, werden automatisch berechenbare Metriken verwendet. Dazu vergleicht man den Ausgabesatz mit einer oder mehreren von Hand generierten Referenzübersetzungen.
Ein mögliches Fehlermaß ist die Levenshtein-Distanz, die definiert ist durch die Anzahl der Auslassungen, Einfügungen und Substitutionen, die man vornehmen muß, um aus dem Ausgabesatz den Referenzsatz zu erhalten. Um lange Sätze nicht zu bestrafen, wird die Gesamtzahl an Fehlern dann über die Gesamtlänge des Referenzsatzes gemittelt.
In dieser Aufgabe schreiben wir ein Programm, das die Levenshtein-Distanz mithilfe von dynamischer Programmierung effizient berechnet.
Daneben gibt es eine Vielzahl an Metriken in der MT-Community, von denen sich vor allem der BLEU-Score durchgesetzt hat. Dieser soll ebenfalls berechnet werden.

Aufgabe:

1. Schreiben Sie ein Programm, das eine Referenzübersetzung $\tilde{e}_1^J$ und eine zu bewertende Übersetzung $\hat{e}_1^K$ einlesen kann.

2. Berechnen Sie die Levenshtein-Distanz:

   - Legen Sie eine Matrix der Länge $J+1$ und der Breite $K+1$ an. In jedem Feld (j,k) soll später die Levenshtein-Distanz für die Strings $\tilde{e}_1^j$ und $\hat{e}_1^k$ stehen.
   - (Initialisierung) Initialisieren Sie die Koordinate $(0,0)$ mit 0.
   - Um zu einem Knoten (j,k) zu gelangen, gibt es drei relevante Übergänge aus vorher berechneten Knoten:
     - eine Löschung von Knoten $(j-1,k)$
     - eine Einfügung von Knoten $(j,k-1)$
     - ein Matching/Substitution von Knoten $(j-1,k-1)$

     Durchlaufen Sie die Matrix in geeigneter Weise und berechnen Sie zu jedem Knoten die minimalen Kosten aus diesen drei möglichen Vorgängerknoten.
   - Lesen Sie aus dem Knoten (J,K) die Levenshtein-Distanz von zwei Sätzen ab.
   - Geben Sie aus, welche Einfügungen, Auslassungen und Ersetzungen ihr Programm in Satz 50 vorgenommen hat, um eine minimale Distanz zu erhalten.

3. Berechnen Sie die positionsunabhängige Levenshtein-Distanz. Berechnen Sie WER und PER für jeden Satz im Testkorpus.

4. Berechnen Sie BLEU:

   - Ein *n-gram* ist eine Folge von n Wörtern. Schreiben Sie eine Funktion, die bei gegebenem n alle Übereinstimmungen von n-grams in der Hypothese mit denen der Referenz zählt. Wird ein n-gram in der Referenz gefunden, gilt es als verbraucht und wird nicht erneut benutzt. Das heißt, zwei identische n-grams in der Hypothese müssen auch zweimal in der Referenz gefunden werden, um voll gezählt zu werden.
   - Berechnen Sie die Präzision p_n einer Hypothese C als
   $$p_n = \frac{\sum_{\text{n-gram} \in C} \text{Count}_{\text{match}}(\text{n-gram})}{\sum_{\text{n-gram}' \in C} \text{Count}(\text{n-gram}')} \tag{1}$$
   - Schreiben Sie eine Funktion, die die sogenannte Brevity Penalty (BP) zurückgibt, wenn c die Länge der Hypothese und r die Länge der Referenz ist:
   $$BP = \begin{cases} 1 & \text{if c} > \text{r} \\ e^{(1-r/c)} & \text{if c} \leq \text{r} \end{cases} \tag{2}$$
   - Berechnen Sie BLEU für $N = 4$ als
   $$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} \frac{1}{N} \log p_n\right). \tag{3}$$

5. Geben Sie für ihre gesamte Übersetzung aus der vorherigen Aufgabe die gemittelte Wortfehlerrate, die gemittelte positionsunabhängige Wortfehlerrate und den BLEU-Score an.

Zusätzliche Aufgaben im Rahmen der Software-Architektur:

- Software-Dokumentation: Dokumentieren Sie die Funktionsweise ihrer Algorithmen und Klassen mit *doxygen*-kompatibler Schreibweise.
- Architektur: Erstellen Sie mit *dot* eine Übersicht über die verwendeten Module und ihrer Ein- und Ausgaberoutinen
- Programmierung: Erstellen Sie ein Profiling mit *gprof* und analysieren/implementieren Sie Verbesserungsmöglichkeiten für die zeitaufwendigsten Teilprozeduren
- Planung: Implementieren Sie Tests, die das korrekte Arbeiten der Verfahren absichern. Etablieren Sie ein Repository, z.B. mit *git* oder *cvs*, und einen Bug-Tracker, z.B. *ditz*

Abnahmetermin: Donnerstag, 10. Juni, ab 14:00 Uhr

Schriftliche Ausarbeitungen werden nicht verlangt. Schicken Sie bitte Ihre kommentierten Quelltexte bereits bis Mittwoch Abend (9. Juni, 18:00 Uhr) an

huck@informatik.rwth-aachen.de.

Am Donnerstag erläutern Sie uns dann Ihre Lösungen und demonstrieren Ihre Programme.