Transliteration Mining

Transliteration wird benutzt, um bspw. russische Namen, die mit kyrillischen Buchstaben geschrieben werden, in lateinischer Schrift darzustellen. Beim **Transliteration Mining** geht es darum, Trainingsdaten für Transliterations-Systeme automatisch aus Korpora zu extrahieren.

In den beiden nächsten Wochen werden Sie das Transliteration-Mining-Modell von Sajjad et al. (`http://www.aclweb.org/anthology/P12-1049`) implementieren.

Als Daten verwenden Sie ein Korpus von Wikipedia Interlanguage Links (WIL). Solche Links verwendet die Wikipedia, um Artikel zum gleichen Thema in unterschiedlichen Sprachen miteinander zu verknüpfen. Bspw. sind die Artikel mit den Titeln *Michail Sergejewitsch Gorbatschow* (de), Mikhail Gorbachev (en), und Горбачёв, Михаил Сергеевич (ru) durch Interlanguage Links verbunden. Ziel ist es, Wortpaare zu finden, die **Transliterationen** voneinander sind. Beispielsweise bilden *Gorbatschow* und Горбачёв ein solches Wortpaar, nicht jedoch *Russland* und Россия.

Sajjad et al. extrahieren zunächst aus den WIL-Daten Wortpaare, die Transliterationspaare sein könnten, und filtern diese "unsauberen" Daten mit einem generativen Modell, welches zwei Teilmodelle vereint: ein *Transliterationsmodell* für die Generierung von Transliterationspaaren und ein *Noisemodell* für die Generierung aller übrigen Wortpaare.

Das **Transliterationsmodell** definiert Wahrscheinlichkeiten für alignierte Transliterationspaare. Ein mögliches Alignment der Wörter *Gorbatschow* und *Gorbachev* sieht wie folgt aus[1].

G:G o:o r:r b:b a:a t: s: c:c h:h o:e w:v

Die Wahrscheinlichkeit eines solchen alignierten Wortpaares $\mathbf{a} = a_1...a_l$ wird als Produkt der Wahrscheinlichkeiten der Transliterationseinheiten a_i (die selbst Buchstabenpaare sind) definiert:

$$p_T(\mathbf{a}) = p_T(a_1...a_l) = \prod_{i=1}^{l} p_T(a_i) \tag{1}$$

Die Wahrscheinlichkeiten $p_T(a_i)$ werden im Training gelernt.

Das **Noisemodell** dagegen generiert zwei Wörter unabhängig voneinander mit zwei monolingualen Teilmodellen. Die Wahrscheinlichkeit eines Wortes $\mathbf{c}$ der Sprache L ist das Produkt der Wahrscheinlichkeiten der einzelnen Buchstaben c_i:

$$p_L(\mathbf{c}) = p_L(c_1...c_n) = \prod_{i=1}^{n} p_L(c_i) \tag{2}$$

$p_L(c_i)$ sind hier die Parameter des Modelles für die Sprache L. Die Noisemodell-Wahrscheinlichkeit des Wortpaares $(\mathbf{x}, \mathbf{y})$ erhalten wir durch Multiplikation der Wahrscheinlichkeiten der beiden monolingualen Modelle für die Sprachen L_1 und L_2:

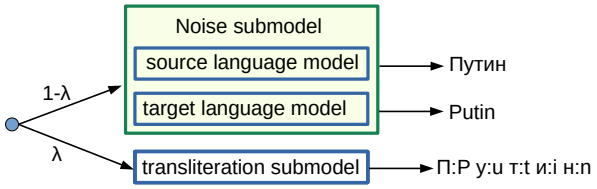
$$p_{noise}(\mathbf{x}, \mathbf{y}) = p_{L_1}(\mathbf{x})\ p_{L_2}(\mathbf{y}) \tag{3}$$

[1] Wegen der besseren Lesbarkeit für nicht-Russischsprecher werden im Folgenden deutsch-englische statt deutsch-russische Beispiele gegeben.

Die Wahrscheinlichkeit, dass das **Transliterationsmodell** das Wortpaar $(\mathbf{x}, \mathbf{y})$ generiert, erhält man durch Summation der Wahrscheinlichkeiten aller möglichen Alignments:

$$p_{trans}(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{a} \in Align(\mathbf{x}, \mathbf{y})} p_T(\mathbf{a}) \tag{4}$$

(Sie werden diese Wahrscheinlichkeit effizienter mit dem forward-Algorithmus berechnen.)



Das **Gesamtmodell** für das Transliteration Mining definiert die Wahrscheinlichkeit eines Wortpaares durch gewichtete Mittelung (=Interpolation) der Wahrscheinlichkeiten des Transliterationsmodelles und des Noisemodelles:

$$p_{mining}(\mathbf{x}, \mathbf{y}) = \lambda \, p_{trans}(\mathbf{x}, \mathbf{y}) + (1 - \lambda) \, p_{noise}(\mathbf{x}, \mathbf{y}) \tag{5}$$

Um die Parameter λ und $p_{trans}(a_i)$ zu lernen, wird das Modell mit dem EM-Algorithmus trainiert. Im Lauf des Trainings passt sich das Transliterations-Teilmodell immer besser an die in den Trainingsdaten enthaltenen Transliterationspaare an, während das (unveränderliche) Noisemodell alle übrigen Wortpaare modelliert. Nach dem Ende des Trainings können die Wortpaare auf Basis der Aposteriori-Wahrscheinlichkeit der Transliteration klassifiziert werden:

$$p(trans|\mathbf{x}, \mathbf{y}) = \frac{\lambda \, p_{trans}(\mathbf{x}, \mathbf{y})}{p_{mining}(\mathbf{x}, \mathbf{y})} \tag{6}$$

Wenn dieser Wert über 0.5 liegt, wird das Wortpaar als Transliteration klassifiziert.

Training

Da die Wikipedia-Titel mehrere Wörter enthalten können und nicht klar ist, welches Wort im Titel der einen Sprache welchem Wort im Titel der anderen Sprache entspricht, werden alle möglichen Paare gebildet. An der Adresse `http://www.cis.uni-muenchen.de/~schmid/lehre/data/wikilinkdata-ru-en.txt.gz` finden Sie Trainingsdaten, bei denen dies bereits erledigt wurde.

Initialisierung: Die Parameter $p_L(c_i)$ jedes **monolingualen Modells** werden einmalig auf allen Wörtern der entsprechenden Sprache L mit relativen Häufigkeiten (ohne Glättung) geschätzt und im Training nicht modifiziert. Die Parameter $p_T(a) = p_T(c_1{:}c_2)$ des **Transliterationsmodells** werden uniform mit $1/(N \cdot M - 1)$ initialisiert, wobei N die Zahl der unterschiedlichen Buchstaben der Quellsprache plus 1 und M die Zahl der unterschiedlichen Buchstaben der Zielsprache plus 1 ist[2]. Der Interpolationsfaktor λ wird mit 0.5 initialisiert.

[2]"plus 1" weil bei Einfügungen (Löschungen) ja das linke (rechte) Zeichen das leere Symbol ist.

EM-Training: Sie machen E-viele EM-Trainingsschritte (E bspw. 3). In jedem Schritt iterieren Sie über alle Trainingsdaten $(\mathbf{x}, \mathbf{y})$ und berechnen mit dem Forward-Backward-Algorithmus die erwartete Häufigkeit f_{trans} von Verwendungen des Transliterationsmodelles (vs. Noise-Modelles) sowie die erwarteten Häufigkeiten $f[c_1{:}c_2]$ der einzelnen Transliterationspaare $c_1{:}c_2$. Nachdem Sie die erwarteten Häufigkeiten im E-Schritt berechnet haben, schätzen Sie im M-Schritt die Parameter λ und die Buchstabenpaar-Wahrscheinlichkeiten $p_T(a) = p_T(c_1{:}c_2)$ neu auf Basis dieser erwarteten Häufigkeiten.

Der **Forward**-Algorithmus berechnet die Wahrscheinlichkeit $\alpha[i,k]$, dass das Transliterations-Modell das i-Präfix von $\mathbf{x}$ und das k-Präfix von $\mathbf{y}$ (mit beliebigem Alignment) zusammen erzeugt.

$$\alpha[i,k] = \alpha[i-1,k]\, p_T(x_{i-1} :) + \alpha[i,k-1]\, p_T(: y_{k-1}) + \alpha[i-1,k-1]\, p_T(x_{i-1} : y_{k-1}) \quad (7)$$

Beachten Sie, dass x_0 und y_0 jeweils den ersten Buchstaben des Wortes bezeichnen, und dass in den Feldern der ersten Zeile (ersten Spalte) nur die Einfüge-Operation (Lösch-Operation) möglich ist.

Der **Backward**-Algorithmus berechnet die Wahrscheinlichkeit $\beta[i,k]$, dass das Transliterations-Modell das Suffix von $\mathbf{x}$ ab Position i und das Suffix von $\mathbf{y}$ ab Position k (mit beliebigem Alignment) zusammen erzeugt.

$$\beta[i,k] = \beta[i+1,k]\, p_T(x_i :) + \beta[i,k+1]\, p_T(: y_k) + \beta[i+1,k+1]\, p_T(x_i : y_k) \quad (8)$$

Beachten Sie, dass in den Feldern der letzten Zeile (letzten Spalte) nur die Einfüge-Operation (Lösch-Operation) möglich ist. Die Werte der Backward-Tabelle werden in umgekehrter Reihenfolge von rechts unten nach links oben berechnet.

Nehmen Sie die Funktion für die Berechnung des Levenshtein-Abstandes aus der letzten Übung als Vorlage für die Implementierung der Forward- und Backward-Wahrscheinlichkeiten und passen Sie sie an.

Nach der Berechnung der Forward- und Backward-Wahrscheinlichkeiten, berechnen Sie zunächst mit Gleichung 5 und 6 die Wahrscheinlichkeit $p(trans|x,y)$, dass das Wortpaar x, y vom Transliterationsmodell generiert wurde. Dazu benötigen Sie $p_{noise}(x,y)$, das Sie mit den monolingualen Modellen berechnen, und $p_{trans}(x,y)$, welches Sie mit dem forward-Algorithmus als $\alpha[n,m]$ berechnet haben.

Nun iterieren Sie noch einmal wie im forward-Algorithmus über alle Felder der Tabelle und berechnen erwartete Häufigkeiten von Buchstabenpaaren mit dem Ausdruck

$$\gamma = p(trans|x,y)\, \frac{\alpha[i-1,k-1]\, p_T(x_{i-1} : y_{k-1})\, \beta[i,k]}{\alpha[n,m]} \quad (9)$$

Den Wert γ addieren Sie zur gesamten erwarteten Häufigkeit $f[x_{i-1}{:}y_{k-1}]$ des Buchstabenpaares $x_{i-1}{:}y_{k-1}$ hinzu. Analog berechnen Sie erwartete Häufigkeiten für $x_{i-1}{:}$ und $:y_{k-1}$. Überlegen Sie, wie Sie dazu Gleichung 9 anpassen müssen.

Für jedes Wortpaar addieren Sie außerdem noch $p(trans|x,y)$ zur erwarteten Häufigkeit f_{trans} von Transliterationen hinzu.

Nachdem Sie die erwarteten Häufigkeiten der Buchstabenpaare und der Transliterationen über alle Wortpaare berechnet haben, schätzen Sie daraus die Buchstabenpaar-

Wahrscheinlichkeiten und den Parameter λ (=Apriori-Wahrscheinlichkeit von Transliteration) neu:

$$p_T(x{:}y) = \frac{f[x{:}y]}{\sum_{z:w} f[z{:}w]}$$
$$\lambda = \frac{f_{trans}}{D}$$

wobei D die Zahl der Wortpaare und $f(x{:}y)$ die erwartete Häufigkeit des Paares x:y ist.

Desambiguierung

Nach dem Training berechnen Sie für alle Wortpaare $(\mathbf{x}, \mathbf{y})$ mit dem trainierten Modell wie im Training die Aposteriori-Wahrscheinlichkeit $p(trans|\mathbf{x}, \mathbf{y})$ von Transliteration. Geben Sie ein Wortpaar $(\mathbf{x}, \mathbf{y})$ aus, falls die Wahrscheinlichkeit $p(trans|\mathbf{x}, \mathbf{y})$ größer als 0.9 ist und es kein anderes Paar (x', y') gibt mit $x' = x$ oder $y' = y$, bei dem die Wahrscheinlichkeit höher ist.

Aufruf: `python extract-transliterations.py word-pair-file.txt`

Die Ausgabe soll auf den Bildschirm gehen.

Hinweise:

- Versuchen Sie, möglichst kurzen und einfachen Code zu schreiben. Überlegen Sie immer wieder, wie Sie Ihr Programm weiter vereinfachen können. Es ist möglich, das Programm mit etwas über 100 Zeilen Code zu implementieren.
- Das Training kann fast eine halbe Stunde dauern. Experimentieren Sie daher zunächst mit einer Teilliste von Wortpaaren, z.B. `http://www.cis.uni-muenchen.de/~schmid/lehre/data/wikilinkdata-ru-en-simp.txt.gz`.

Vorüberlegungen

- Wie können Sie die Backward-Wahrscheinlichkeiten mit Hilfe des Forward-Algorithmus implementieren?
- Wie können Sie zum Schluss die besten Transliterationspaare effizient berechnen?
- Welche Klassen verwenden Sie, um den Code sinnvoll zu strukturieren und welche Interface-Methoden (öffentliche Methoden) sollten die Klassen bereitstellen?

  Definieren Sie eine Klasse `NoiseModel` *mit den Dictionary-Attributen* `src_letter_prob` *und* `tgt_letter prob` *(für $p_{L_1}(letter)$ und $p_{L_2}(letter)$) und dem Konstruktor* `NoiseModel(word_pair_list)`*, welcher die Häufigkeiten der Buchstaben berechnet und damit Ihre Wahrscheinlichkeiten schätzt, und der Methode* `word_pair_prob(src_word, tgt_word)`*, welche die Wahrscheinlichkeit von* `src_word` *und* `tgt_word` *berechnet und multipliziert.*

  Außerdem definieren Sie eine Klasse `TransliterationModel` *mit Dictionary-Attributen* `trans_unit_prob` *(für die Wahrscheinlichkeiten der Buchstabenpaare a:B) und* `trans_unit_freq` *(zum Aufsammeln der erwarteten Buchstabenhäufigkeiten), dem Konstruktor* `TransliterationModel(src_letters, tgt_letters)`*, welcher das Transliteration-Modell initialisiert, und den Methoden* `add_freq(src_letter, tgt_letter, freq)`*, welche den Wert* `freq` *zur erwarteten Häufigkeit des Paares*

`src_letter:tgt_letter` addiert, und `reestimate()`, welche die Buchstabenpaar-Wahrscheinlichkeiten aus den erwarteten Häufigkeiten ohne Glättung neuschätzt.

Schließlich brauchen Sie noch die Hauptklasse `MiningModel` mit den Attributen `noise_model`, `trans_model`, `trans_prior` (für den Parameter λ), `trans_freq` (für die erwartete Häufigkeit der Verwendung des Transliterationsmodelles) und dem Konstruktor `MiningModel(word_pair_list)`, welcher die Teilmodelle initialisiert und das Training startet. Dann brauchen Sie noch eine Methode `estimate_freqs(word_pair_list)`, welche für jedes Wortpaar die Methoden `forward(src_word, tgt_word)` und `backward(src_word, tgt_word)` aufruft und die erwarteten Buchstabenpaar-Häufigkeiten berechnet und aufsummiert, und eine Methode `reestimate_probs()` für den M-Schritt, eine Methode `em(word_pair_list, n)`, welche n-mal den E-Schritt und den M-Schritt ausführt, und eine Methode `print_transliterations(word_pair_list)`, welche nochmals über alle Wortpaare (x,y) iteriert und diejenigen ausgibt, die als Transliterationspaare klassifiziert werden.
*Vergessen Sie nicht, die Häufigkeitstabellen nach dem M-Schritt wieder zu initialisieren. In der Methode `estimate_freqs` müssen Sie wie in der Methode `forward` über **alle** Felder der Matrix iterieren.*

Im Hauptprogramm werden die Klassen so verwendet:

```
model = MiningModel(word_pair_list, num_iterations)
model.print_transliterations(word_pair_list)
```

Mit dem Befehl `src_words, tgt_words = zip(*word_pair_list` können Sie zwei Listen mit den Source- bzw. Target-Wörtern erzeugen.