A2, Umsetzung, Abschlussbericht

Abdullah, Koukash, abk9550@thi.de, Gruppe 1

6. November 2022

1 Einleitung

In dieser Aufgabe geht es darum, ein buchstabenbasierten n-gramm Sprachmodell zur Sprachen-Erkennung zu entwerfen. Dabei kamen der Naïve Bayes-Klassifikator und Laplace (add-1) smoothing (Glättung) zum Einsatz. Wie und warum, ist in den folgenden Abschnitten zu erfahren.

2 Datensatz

Ich habe den Datensatz „WiLI-2018“ benutzt, welcher uns zur Verfügung gestellt wurde. „WiLI-2018“ ist ein großer Datensatz, welcher einen Test- und Trainingsdatensatz für 235 verschiedene Sprachen und Dialekte beinthaltet. es sind genau 117500 Trainingsdaten und 117500 Testsdaten. Für jede Sprache/Dialekt jeweils 500 Texte (für das Testen 500, für das Trainieren 500). „WiLI-2018“ ist ein riesig großer Datensatz, welcher bei dem die Vorverarbeitung aufwendig ist, welcher aber Vorteile mit sich bringt, Bspw. dass das Sprachmodell für das Training möglichst viele Daten hat und somit die Gefahr, ein „data sparsity“-Problem zu haben, zu verringern [1]

3 Training

3.1 Texte Vorverarbeitung

Zuallererst müssen wir die Daten für das Training vorverarbeiten.

1. Die Zeichensetzungen und Ziffern entfernen {%$&#ˆ+*-:;_˜.,!?@ [‘’]() „“ 0-9}
2. Die unnötigen Leerzeichen entfernen
3. den Text in Kleinschreibung umwandeln

Das macht die Funktion: **text_vorverarbeiten(text)**, dann durch werden die n-gramme erstellt.

3.2 Naïve Bayes-Klassifikator & Laplace (add-1) smoothing

3.2.1 Naïve Bayes-Klassifikator

Nach einer Langen Recherche hat sich ergeben, dass der Naïve Bayes-Klassifikator am besten für die Problemstellung geeingt ist [3]. Die naïve Bayes-Klassifikatorsformel, wobei L für die Sprache steht, $\underline{W}$ für eine n-gramm Folge steht:

$$LID\left(\underline{W}\right) = argmax_{L\{de,en,...\}}\ P\left(L|\underline{W}\right)$$

durch den Satz von Bayes können wir die Formel umdrehen:

$$P\left(L|\underline{W}\right) = argmax_{L\{de,en,...\}}\ P\left(\underline{W}|L\right)P\left(L\right)$$

Die Frage ist jetzt: Wie berechnet man $P\left(\underline{W}|L\right)$ und $P\left(L\right)$?

3.2.2 Laplace (add-1) smoothing & $P(\underline{W}|L)$ & $P(L)$

Man könnte den Naïve Bayes-Klassifikator ohne smoothing anwenden, ich habe mich aber bewusst dafür entschieden, den Klassifikator mit Laplace (add-1) smoothing zu nutzen. Ich habe gemerkt, dass durch die Glättung, bessere Ergebnisse bei manchen Sprachen erzielt werden könnten bzw. die Glättung führte zu besseren Schätzungen. Außerdem will ich ungern 0-Wahrscheinlichkeiten vergeben, weil 0-Wahrscheinlichkeiten im Likelihood-Term für eine beliebige Sprache führen dazu, dass die Wahrscheinlichkeit der Sprache 0 ist [4]. Die Formel sieht dann so aus: $P(\underline{W}|L) = \prod_{i=1}^{n} \frac{count(\ W_i,L)+1}{count(L)+|V|}$
Wobei:

- $count\,(\ W_i, L)$ wie oft ein n-gramm in einer Sprache vorkommt
- $count\,(L)$ die Anzahl aller n-gramme in der Sprache
- $|V|$ die Anzahl aller n-gramme über alle Sprachen (Vokabular)

$P(L)$ ist gewöhnlich gleichverteilt $\frac{1}{|L|}$ (wobei $|L|$ Anzahl der Sprachen ist). Es kommt aber auf die Anwendung drauf an [5]. Ich habe mich für die Gleichverteilung entschieden.

3.2.3 Log-Wahrscheinlichkeit

Es ist dem Rechner aufwendig und es ist auf CPU numerisch instabil was die Genauigkeit angeht, das $P(\underline{W}|L) = \prod_{i=1}^{n} \frac{count(\ W_i,L)+1}{count(L)+|V|}$ zu berechnen. weil je länger $\underline{\mathrm{W}}$ ist, desto geht die Wahrschinlichkeit gegen null.
Deshalb wird die log-Wahrscheinlichkeit benutzt, weil 1. $log_2(x * y) = log_2(x) + log_2(y)$, Addition kann der Rechner viel schneller als die Multiplikation berechnen und 2. die Genauigkeit ist höher und nummerisch stabiler auf CPU [3][5]. Also die neue Formel: $P(\underline{W}|L) = \sum_{i=1}^{n} log_2(\frac{count(\ W_i,L)+1}{count(L)+|V|})$

4 Evaluation

4.1 Testdatensatz & Vorbereitung

Wie bereits erwähnt, enthält der „WiLI-2018“ Datensatz bereits einen Testsdatensatz mit 117500 verschiedene Texte, pro Sprache sind es 500 Texte zum testen. Ich habe den kompletten Testsdatensatz vorverarbeitet und dann durch das Modell laufen lassen. Ich werde die Ergebnisse, für einen besseren Überblick, nur für die wichtigsten 13 Sprachen darstellen (für die Ergebnisse aller Sprachen siehe Jupyter-Notebook). Durch die Funktion **schaetze(text)** lassen sich für den übergegebenen Parameter „text“ die 3 am besten passenden Sprachen ausgeben.

4.2 Ergebnisse

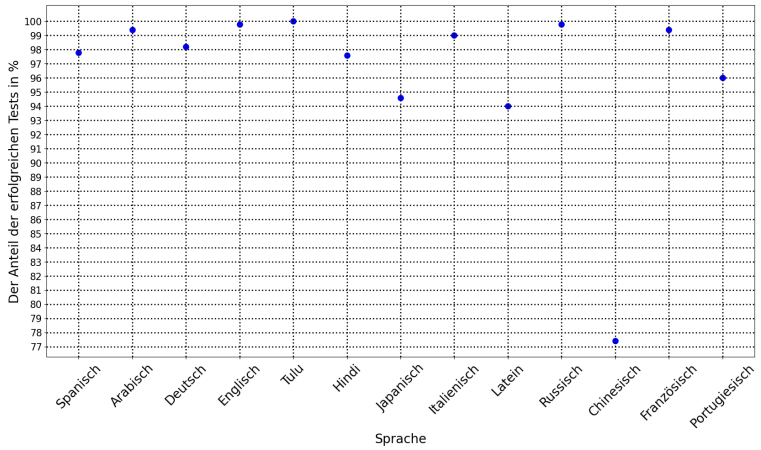

Abbildung 1: Testergebnisse mit buchstabenbasierten 4-gramm Sprachmodell

4.3 Interpretation der Ergebnisse

Wir können in der Abbildung 1 sehen, dass alle Sprachen außer Chinesisch eine Genuaigkeit über 94% haben. die Genauigkeit der chinesische Sprache liegt bei 77.4%. Das ist ein guter Wert aber es könnte besser sein. Die chinesische Sprache ist in NLP im allgemeinen schwer zu verarbeiten [6]. Außerdem könnte es auch daran liegen, dass wir mehr Datensätze für die chinesische Sprache brauchen, denn mehr Daten wären niemals von Nachteil. Die durchschnittliche Genauigkeit über alle Sprachen (235 Sprachen) liegt bei **96.95%**, was ein sehr guter Wert ist.

5 Verbesserungsmöglichkeiten

Es wäre nicht schlecht, verschiedene Tests auszuführen, um zu sehen welches n-gramm Modell, die besten Ergebnisse erzielt. Ich habe es leider nur mit einem 4-gramm Sprachmodell getestet, weil die Tests sehr aufwendig waren.
Ein Schwachpunkt meiner Entwicklung wäre es, wenn ein einziger Text mehrere Sprachen enthält.
Verbesserungsmöglichkeiten wären, 1. Eine bessere Textvorverarbeitung durchzuführen (Bspw. durch Stemming und lemmatization). 2. Ich würde aus diesem Sprachen-Detektors einen Multi-linguale Sprachen-Detektors entwerfen.

6 Referenzen

[**1**] Data sparsity: ist der Begriff, der verwendet wird, um das Phänomen zu beschreiben, nicht genügend Daten in einem Datensatz zu beobachten. https://www.sciencedirect.com/science/article/abs/pii/S1568494617303071
[**2**] https://www.cs.rhodes.edu/~kirlinp/courses/ai/f18/projects/proj3/naive-bayes-log-probs.pdf
[**3**] https://www.cc.gatech.edu/home/isbell/classes/reading/papers/Rish.pdf
[**4**] https://web.stanford.edu/~jurafsky/slp3/ed3book_jan122022.pdf Abschnitt 4
[**5**] Vorlesung
[**6**] https://www.researchgate.net/profile/Kam-Fai-Wong/publication/2872059_Application_and_Difficulty_of_Natural_Language_Processing_in_Chinese_Temporal_Information_Extraction/links/581085b408ae009606be3099/Application-and-Difficulty-of-Natural-Language-Processing-in-Chinese-pdf