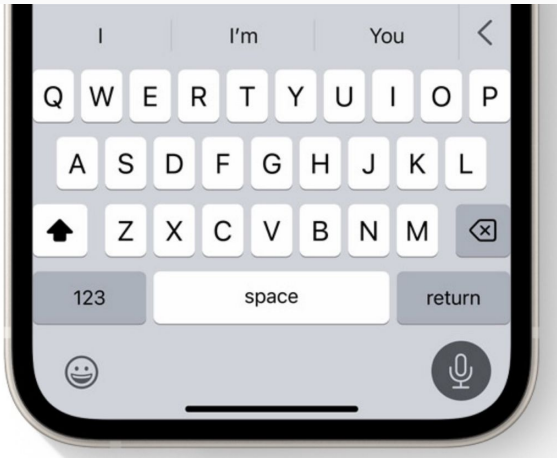
Implementera en ordprediktor

Anton Johansson & Nathalie Haghshenas

Uppgiftsbeskrivning för ordprediktorn

- Ordprediktor likt de som finns i telefoner, uppdateras vid varje knapptryck
- Eventuellt föreslå rättningar och/eller lära sig nya uttryck användaren nyttjar



"En bra ordprediktor använder ord-n-gram-sannolikheter för att kunna föreslå den troligaste kompletteringen"

Bakgrund 1 - Statistiska språkmodeller & Markov-antagande

- Antar att ord är slumpmässiga och genereras av stokastisk process
- Genererar sannolikheter av ords förekomst givet kontext (föregående ord)
- Kan användas för prediktion, stavningskontroll, översättning m.m.
- Begränsar kontext genom Markov-antaganden
- "Assume that a word only depends on the previous couple of words"

*Kallas även stokastisk/probabilistisk språkmodell

P(I really like ants) =

P(ants|I really like)P(like|I really)P(really|I)P(I)

N-gram-modeller tar de föregående n-1 orden i beaktande vid förutsägande av nästa ord:

Unigram: P(I)P(really)P(like)P(ants)

Bigram: P(I)P(really|I)P(like|really)P(ants|like)

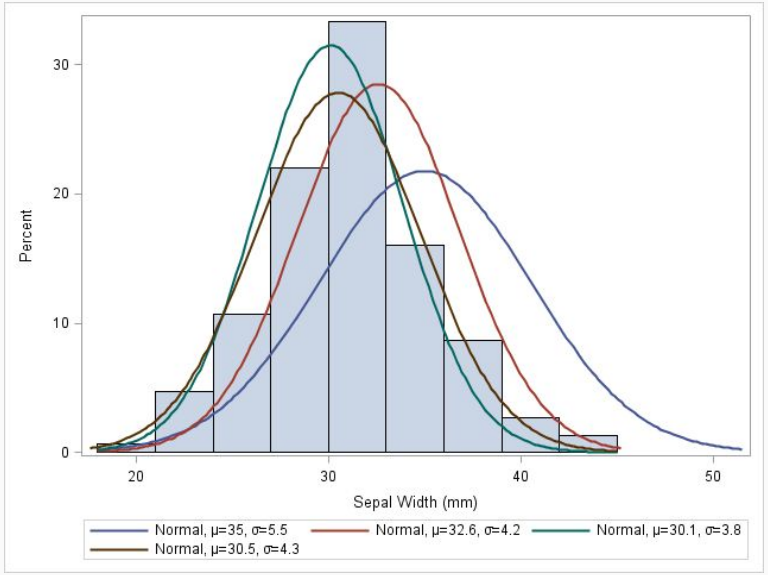
$$P(w_i|w_1,\ldots,w_{i-1}) = P(w_i|w_{i-n+1},\ldots,w_{i-1})$$

Bakgrund 2 - MLE

Beräkna sannolikheten av olika ords förekomst i kontext

P(to|like) = C(like to)/C(like)

$$P(w_i|w_{i-n+1},\ldots,w_{i-1}) = \frac{c(w_{i-n+1},\ldots w_i)}{c(w_{i-n+1},\ldots w_{i-1})}$$



Bakgrund 3 - Felstavning & Levenshtein

- Ger en “kostnad” för insättning, borttagning eller ersättning.
- Manipulerar ett ord mot mål-ordet
- Summan av kostnaderna är avståndet

		m	e	i	l	e	n	s	t	e	i	n
	0	1	2	3	4	5	6	7	8	9	10	11
l	1	1	2	3	3	4	5	6	7	8	9	10
e	2	2	1	2	3	3	4	5	6	7	8	9
v	3	3	2	2	3	4	4	5	6	7	8	9
e	4	4	3	3	3	3	4	5	6	6	7	8
n	5	5	4	4	4	4	3	4	5	6	7	7
s	6	6	5	5	5	5	4	3	4	5	6	7
h	7	7	6	6	6	6	5	4	4	5	6	7
t	8	8	7	7	7	7	6	5	4	5	6	7
e	9	9	8	8	8	7	7	6	5	4	5	6
i	10	10	9	8	9	8	8	7	6	5	4	5
n	11	11	10	9	9	9	8	8	7	6	5	4

- Ord som inte förekommer i vårt corpus, C(w) = 0 kan vi anta är felstavade*
- Tillsammans med dess sannolikhet att förekomma i kontexten kan vi ge användare förslag på “rätt” ord

*Alternativt låtit användaren utöka corpus med ordet

Snabbt om Hypotes

- Trigram > bigram
- Textcorpus begränsande
- Troligtvis innehåller slang
- Större träningscorpus → mer träffsäker prediktion

Implementation 1: Återanvänt

- BigramTrainer → TrigramTrainer
- Utökades för att beräkna trigramsannolikheter och skapa vår språkmodell utifrån ett corpus
- Beräknar uni-, bi- och trigrams-sannolikheter genom MLE

- BigramTester → TrigramTester
- Utökades för att kunna utvärdera språkmodellen.
- TrigramTester beräknar kors-entropin mellan modell och en testmängd
- Summera modellens entropi för varje enskilt ord.

Implementation 2: Nyskapat

- Prediktorn = Predictor.py.
- Enstaka likheter i struktur med Generator.py från labb 2
- GUI i klass “GuiWindow”. Knapptryck binds till att kalla på särskilda metoder i prediktorn. Deras innehåll samt användarens inmatningspanel uppdateras efter tangent-tryck eller knapptryck
- Prediktor i klass “Predictor”. Innehåller metoder för att (förutom att läsa in modell) generera nya ord beroende på kontexten. Olika metoder beroende på antalet tidigare ord (Backoff) och huruvida användaren skriver på ett ord eller inte

Backoff - en teknik där man, om en sannolikhet från en högre ordnings n-gram-modell inte är tillgänglig, väljer att använda sannolikheter från en n-gram-modell av lägre ordning

Data - “Mobile text dataset”

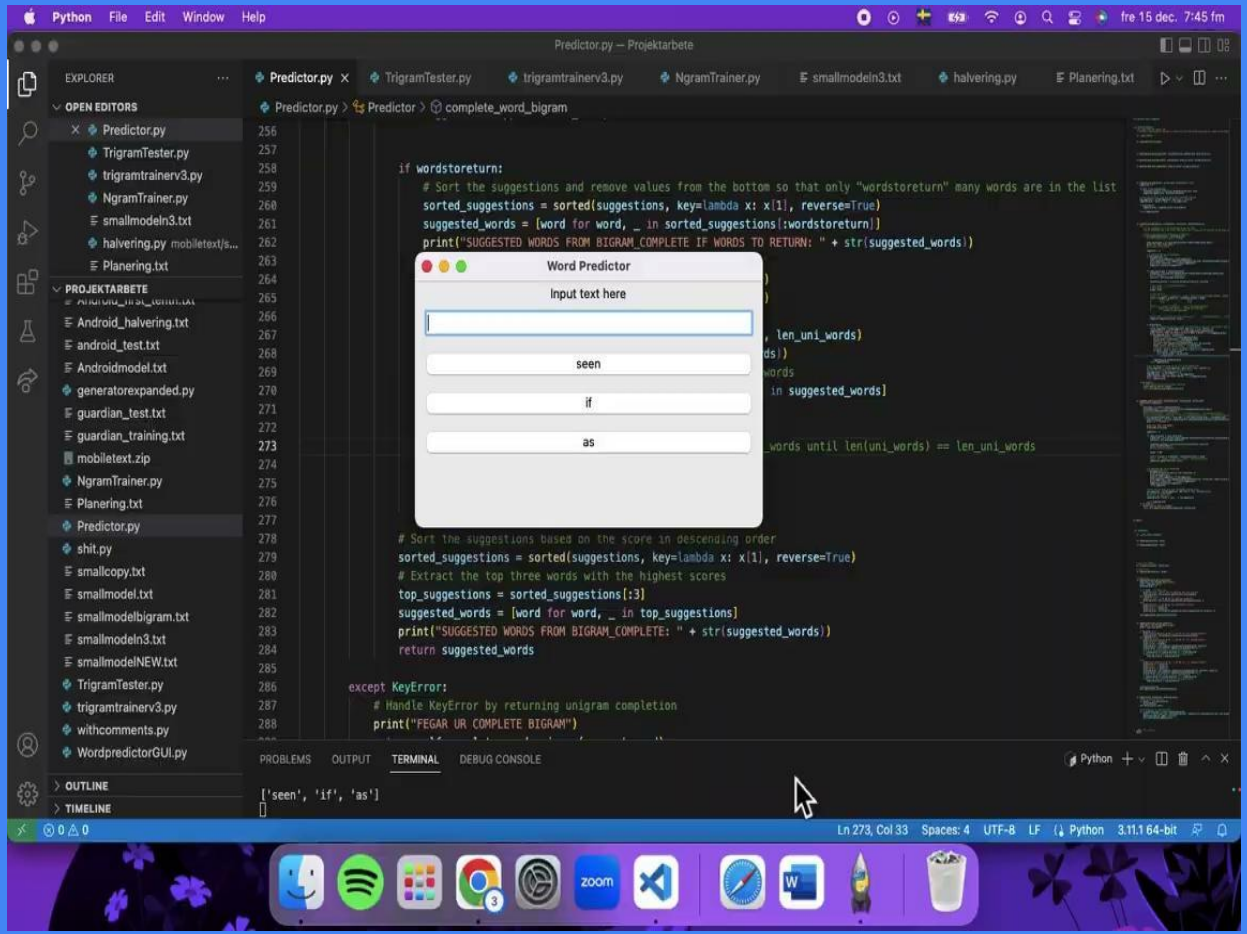
Hämtat från studien “Mining, analyzing, and modeling text written on mobile devices” av Kristensen & Vertanen, 2021

En studie från Cambridge & Michigan Technical University där forskarna försöker tillgängliggöra data för studier om hur vi skriver på telefoner och andra små enheter, genom web scraping och utsållning genom enheters signaturer

Resultatet är flertalet txt-filer innehållandes inlägg som användare gjort på diverse forum/medier från olika typer av enheter, bl.a. Tränings- utvecklings- och evaluerings-set för språkmodellering samt en intressant diskussion gällande användarvanor och språkmodellers (bristande) träffsäkerhet

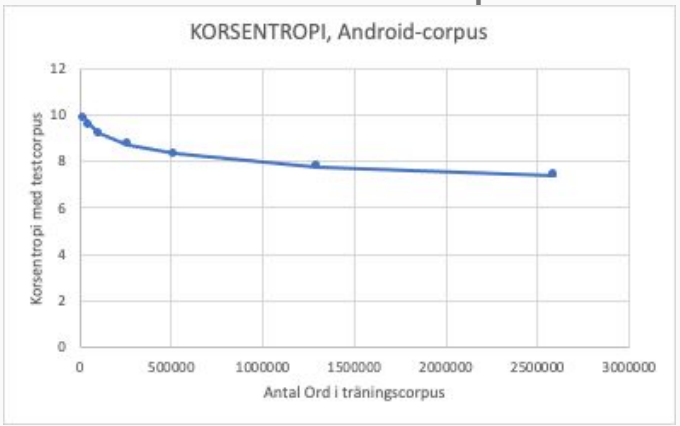
Vertanen K, Kristensson PO. Mining, analyzing, and modeling text written on mobile devices. *Natural Language Engineering*. 2021;27(1):1-33. doi:10.1017/S1351324919000548

DEMO

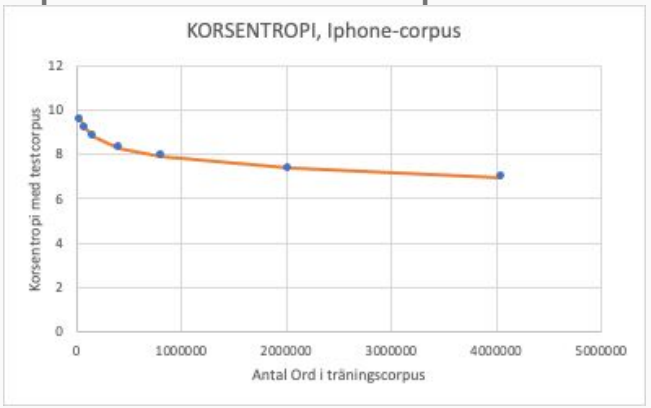


Resultat - Korsentropi med test-set i TrigramTester

Android-andvändar-corpus:



Iphone-användar-corpus



- Modellen blir mer träffsäker med större corpus
- Fortfarande relativt hög entropi, beror med stor sannolikhet (delvis) på testprogrammet

Slutsats

Modellen följer de förväntningar vi har (om än med hög entropi) på statistiska språkmodeller

Antagandet om ord som slumpmässiga, stokastiskt uppkommande fenomen ignorerar grammatik och begränsar starkt prediktorns förmåga att ge användbara rekommendationer

Bättre lösning med Levenshteinavstånd eller annan metod för stavningskontroll/ordkomplettering bör finnas