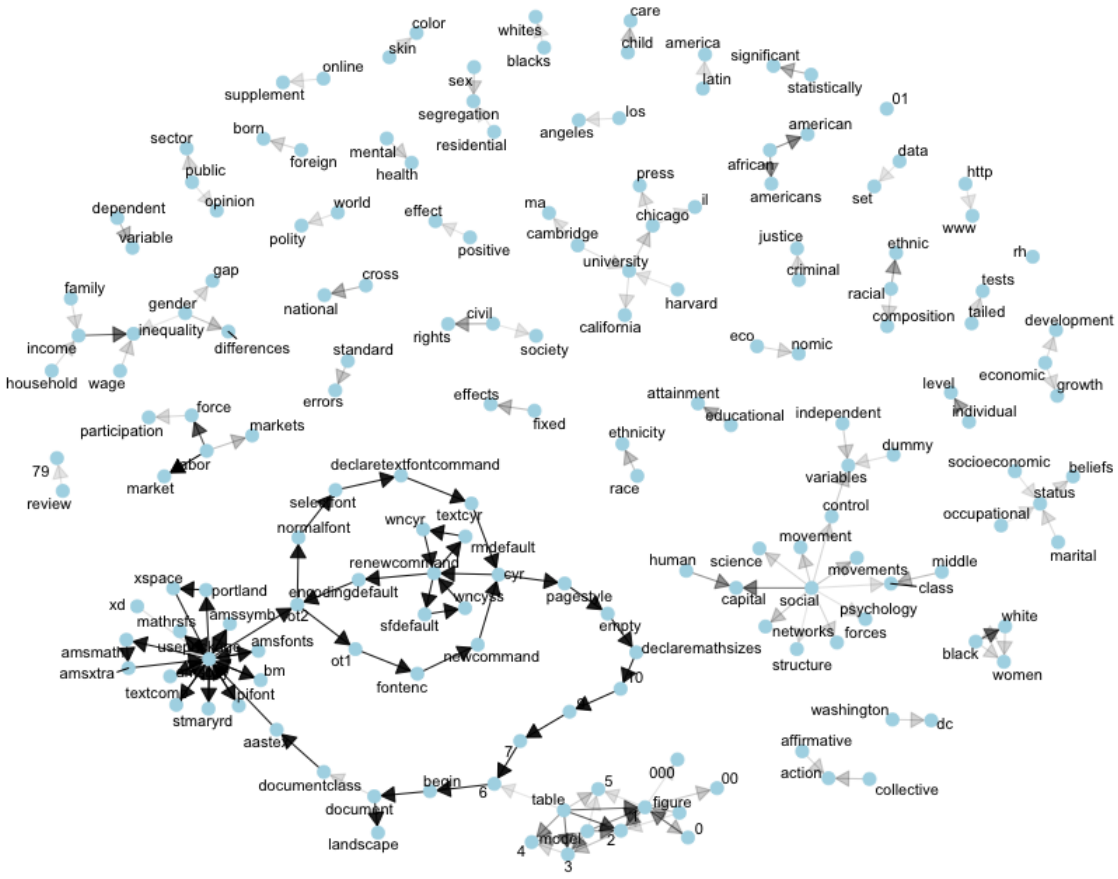
Onderzoeksrapport inleveropdracht 3



Max Goldsmits & Justin Schots
1836919, 1812181
ZM-0006 - AI, Zuyd Hogeschool

Inhoud

Inleiding

Het n-gram model is een veelgebruikte techniek in natuurlijke taalverwerking (NLP) en tekstanalyse. N-grams zijn continuïteitsmodellen voor tekst die een reeks van n opeenvolgende items in een tekst representeren, waarbij een item meestal een woord is. Laten we een uitgebreide uitleg geven over het n-gram model, inclusief een vergelijking tussen bigrams (bi-grams) en trigrams (tri-grams), en een beschrijving van alternatieven, zoals bag-of-words en continuïteitsmodellen.

1. N-gram Model:

Het n-gram model is een taalmodel dat de waarschijnlijkheid van een bepaalde reeks van n opeenvolgende woorden in een tekst modelleren. Dit model is gebaseerd op het principe van Markovketens, waarbij de kans op een bepaald woord afhankelijk is van de voorgaande n-1 woorden. Het model is gebaseerd op de veronderstelling dat de kans op een woord sterk afhankelijk is van de context die door de voorgaande n-1 woorden wordt geboden. (Jurafsky & Martin, 2023)

Hier is een voorbeeld van een trigram (n = 3) met de zin "Ik hou van natuurlijke taalverwerking." Het trigrammodel zou de kans berekenen dat het volgende woord "taalverwerking" is, gegeven de voorgaande twee woorden "hou van" en "natuurlijke."

2. Bigrams (Bi grams) vs. Trigrams (Trigrams):

Bigrams (Bi-grams): In een bigrammodel worden reeksen van twee opeenvolgende woorden gemodelleerd. Dit betekent dat de kans op een woord afhangt van het voorgaande woord. Bigrams zijn eenvoudiger dan trigrams en kunnen nuttig zijn voor taken zoals tekstclassificatie en tekstvoorspelling, maar ze hebben de neiging minder contextuele informatie vast te leggen.

Trigrams (Tri-grams): Trigrammodellen omvatten reeksen van drie opeenvolgende woorden, waardoor ze meer contextuele informatie vastleggen. Dit maakt trigrams krachtiger voor taken waarbij diepere semantische betekenis en context vereist zijn. Ze worden vaak gebruikt in automatische spraakherkenning en machinevertaling. (Mattingly, 2022)

3. Alternatieven voor N-grams:

3.1 Bag-of-Words (BoW):

Het Bag-of-Words-model is een eenvoudige en veelgebruikte techniek in NLP. Het idee achter BoW is om tekst te behandelen als een verzameling van losse woorden, zonder rekening te houden met de volgorde waarin ze voorkomen. Hieronder is in stappen opgedeeld hoe dit werkt:

Vocabulaire: Eerst wordt een vocabulaire gecreëerd, bestaande uit alle unieke woorden in de gegeven teksten. Deze woorden worden tokens genoemd.

Woordfrequentie: Voor elk document wordt bijgehouden hoe vaak elk woord uit het vocabulaire voorkomt in dat document. Dit leidt tot een vector die de frequentie van elk woord in het document weergeeft.

Documentrepresentatie: Elk document wordt uiteindelijk gerepresenteerd als een vector waarvan de dimensies overeenkomen met het vocabulaire. De waarde van elke dimensie is de frequentie van het overeenkomende woord in het document.

BoW is eenvoudig te implementeren en geschikt voor taken zoals tekstclassificatie en sentimentanalyse. Het belangrijkste nadeel is dat het de volgorde van woorden negeert en daardoor veel contextuele informatie verliest. Bijvoorbeeld, zinnen met dezelfde woorden in verschillende volgordes zouden dezelfde BoW-representatie hebben. (Great Learning Team, 2022)

3.2 Word Embeddings:

Word Embeddings zijn een geavanceerde techniek om woorden in een continue ruimte te plaatsen, waarbij semantische relaties tussen woorden worden vastgelegd. In tegenstelling tot BoW houden Word Embeddings wel rekening met de semantische betekenis van woorden en hun onderlinge relaties.

Continue Vectoren: Elk woord in het vocabulaire wordt gemapt naar een continue vector in een meerdimensionale ruimte. Deze vectoren bevatten semantische informatie, wat betekent dat woorden met vergelijkbare betekenis dicht bij elkaar in deze ruimte liggen.

Word2Vec en GloVe: Word2Vec en GloVe zijn populaire algoritmen voor het genereren van Word Embeddings. Ze leren deze vectoren vanuit grote tekstcorpora door te proberen woorden te voorspellen op basis van hun context in zinnen.

Semantische Betekenis: Door het gebruik van Word Embeddings kunnen NLP-modellen semantische betekenis begrijpen. Ze zijn nuttig voor veel NLP-taken, zoals taalvertaling, sentimentanalyse, tekstklassificatie en aanbevelingssystemen.

Het voordeel van Word Embeddings is dat ze de semantische betekenis van woorden vastleggen en de context behouden. Hierdoor kunnen NLP-modellen beter begrijpen hoe woorden in zinnen en teksten samenwerken. (Karani, 2018)

3.3 Recurrent Neural Networks (RNNs):

Recurrent Neural Networks zijn een klasse van neurale netwerken die zijn ontworpen om de volgorde van woorden in teksten vast te leggen en sequentiële informatie te modelleren.

Sequentiële Informatie: RNN's hebben een speciale architectuur waarmee ze informatie kunnen doorgeven van de ene tijdstap (woord) naar het volgende. Hierdoor kunnen ze complexe sequentiële patronen vastleggen.

Toepassingen: RNN's worden vaak gebruikt voor taken waarbij de volgorde van woorden belangrijk is, zoals automatische spraakherkenning, tekstanalyse, machinevertaling en chatbots.

Het voordeel van RNN's is hun vermogen om complexe sequentiële patronen vast te leggen. Ze kunnen worden gebruikt om diepere context en taalbegrip vast te leggen. Een van de beperkingen is dat ze gevoelig zijn voor het verdwijnen van gradiënten, wat training bemoeilijkt.

In vergelijking met BoW en Word Embeddings kunnen RNN's de volgorde en context van woorden in teksten beter vastleggen, maar ze zijn complexer om te trainen en vereisen meer rekenkracht. De keuze tussen deze technieken hangt af van de specifieke vereisten van een NLP-taak. (IBM, sd)

3.4 Vergelijking tussen Alternatieven en N-grams:

De keuze tussen verschillende benaderingen in natuurlijke taalverwerking (NLP) hangt sterk af van de specifieke behoeften van een taak. Hieronder volgt een uitgebreidere vergelijking tussen N-grammodellen, Bag-of-Words (BoW), Word Embeddings en Recurrent Neural Networks (RNNs), rekening houdend met de voor- en nadelen van elk en met verwijzing naar relevante bronnen.

N-grammodellen: N-grammodellen zijn zeer geschikt voor taken waarbij de volgorde van woorden en context cruciaal zijn. Ze kunnen nauwkeurige voorspellingen doen op basis van de voorgaande n-1 woorden. Bijvoorbeeld, ze kunnen goed presteren in taalmodellering en tekstanalyse waarbij de contextuele afhankelijkheden tussen woorden van belang zijn. Een voorbeeld van een bron die N-grammodellen behandelt, is "Speech and Language Processing" door Jurafsky en Martin (2019).

Bag-of-Words (BoW): BoW is een eenvoudige en efficiënte aanpak voor NLP-taken waarbij context minder belangrijk is. Het is gemakkelijk te implementeren en kan goed werken voor taken zoals tekstclassificatie en informatie-extractie. Echter, BoW verliest veel contextuele informatie en houdt geen rekening met de volgorde van woorden. Dit maakt het minder geschikt voor taken waarbij de volgorde en diepere semantische betekenis van de tekst van belang zijn. Het boek "Introduction to Information Retrieval" door Manning, Raghavan, en Schütze (2008) biedt inzicht in BoW-technieken.

Word Embeddings: Word Embeddings, zoals Word2Vec en GloVe, leggen semantische relaties tussen woorden vast en zijn geschikt voor semantische taaltaken. Ze kunnen een diepgaand begrip van de betekenis van woorden bieden en kunnen bijvoorbeeld sentimentanalyse en semantische zoekopdrachten verbeteren. Een voorbeeld van een bron die Word Embeddings behandelt, is het artikel "Efficient Estimation of Word Representations in Vector Space" door Mikolov et al. (2013).

Recurrent Neural Networks (RNNs): RNN's zijn ideaal voor taken waarbij de volgorde van woorden van groot belang is, zoals machinevertaling, tekstanalyse en spraakherkenning. Ze zijn in staat om complexe sequentiële patronen vast te leggen en bieden diepere contextuele inzichten. Het artikel "Long Short-Term Memory" door Hochreiter en Schmidhuber (1997) beschrijft een populaire vorm van RNN genaamd Long Short-Term Memory (LSTM).

In de praktijk kan de keuze tussen deze technieken afhankelijk zijn van factoren zoals de beschikbaarheid van gegevens, de aard van de taak en de gewenste nauwkeurigheid. N-grammodellen zijn geschikt voor taken waarbij de contextuele afhankelijkheden tussen woorden van belang zijn, terwijl alternatieven zoals BoW en Word Embeddings geschikt zijn voor situaties waarin minder contextuele informatie nodig is of waar computationale beperkingen een rol spelen. RNNs, hoewel krachtig, vereisen vaak voldoende gegevens en rekenkracht voor training en worden meestal toegepast op complexe taaltaken waar de volgorde van woorden van cruciaal belang is.

Bronnen:

Jurafsky, D., & Martin, J. H. (2019). Speech and Language Processing (3rd ed.). Prentice-Hall.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.

Mikolov, T., et al. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.

4. Beschrijving van het programma

Onze opgeleverde code is een programma dat bedoeld is om de taal van een gegeven zin te detecteren op basis van bi- en trigrammen. Het maakt gebruik van verschillende taalmodellen en scripts in verschillende talen om deze detectie uit te voeren.

4.1 Datasets en bronnen:

De datasets die in ons programma worden gebruikt, zijn de scripts van verschillende films in verschillende talen. Deze scripts bevatten tekst in zowel de originele taal van de film als enkele andere talen. De scripts die worden gebruikt, zijn afkomstig uit de films "Titanic," "GhostBusters," "TheCureForInsomnia," en "WolfOfWallStreet." Deze scripts zijn geselecteerd omdat ze een breed scala aan woorden en uitdrukkingen bevatten en de mogelijkheid bieden om taal te herkennen op basis van de tekst.

4.2 Training:

Ons programma is niet getraind in de traditionele zin van machine learning. In plaats daarvan maakt het gebruik van de teksten in de scripts om bi- en trigrammen te genereren voor verschillende talen. Het programma slaat deze n-grammen op in dictionaries voor elke taal. Dit is een heuristische benadering om te bepalen welke n-grammen vaak voorkomen in de scripts van verschillende talen.

4.3 Opslag:

De bi- en trigrammen worden opgeslagen in dictionaries, waarbij de taal als sleutel fungeert en de n-grammen als waarden. Deze dictionaries worden opgebouwd tijdens de uitvoering van het programma.

4.4 Kansberekening per bi- en trigram:

De kansberekening per bi- en trigram wordt gedaan door te tellen hoe vaak een gegeven n-gram voorkomt in de inputzin en te vergelijken met de opgeslagen n-grammen in de dictionaries voor elke taal. De taal met de hoogste overeenkomst wordt beschouwd als de gedetecteerde taal.

4.5 Smoothing:

Ons programma maakt geen gebruik van smoothing-technieken, zoals Laplace-smoothing. In plaats daarvan gebruikt het gewoon de frequentie van de n-grammen in de scripts. Dit betekent dat als een bepaald n-gram niet in een script voorkomt, het gewicht voor die taal op nul wordt gezet, wat betekent dat dit n-gram geen invloed heeft op de detectie van die taal.

5. Kwantitatieve evaluatie van het algoritme

In deze kop wordt de vergelijking tussen bi- en tri-grammen gesteld op basis van de correctheid en de performance van het geschreven algoritme. Onder de koppen staat figuur 1. Hierin is de console output met elke benodigde metriek verzorgt.

5.1 Correctheid

De correctheid van de algoritmes is onafhankelijk van elkaar. Zo heeft een bigram weinig woorden in een zin nodig om een taal te herkennen uit een ingevoerde zin tegenover een tri-gram. Uit het testen van het eigengemaakte algoritme bleek dat een tri-gram rond de 9 woorden in een zin nodig heeft om het correcte te detecteren tegenover de minimaal 3 voor een bigram.

5.2 Performance

Het algoritme wordt heel snel uitgevoerd, dit is zo omdat de code op een manier geoptimaliseerd is met functies die elkaar aanroepen wanneer dit nodig is in de code. Op deze manier worden er geen computerkracht verspild. Dit valt ook te zien aan de code wanneer deze wordt uitgevoerd. Voor het berekenen van de runtime is de time import in python gebruikt.

```
Enter a sentence: En una calurosa tarde de verano, mientras el viento cálido acariciaba suavemente los campos de trigo dorado y las
cigarras cantaban en coro, un grupo de amigos decidió aventurarse en un largo viaje a través de la selva tropical, explorando la exu
berante vegetación, descubriendo cascadas ocultas y maravillándose ante la diversidad de la flora y la fauna que habita en este herm
oso rincón del mundo, donde la naturaleza revela su esplendor en cada rincón, y el tiempo parece detenerse en medio de tanta belleza
The input sentence is written in Spanish according to the bigrams.
Language   Score      %
English    1          3.03
French     2          6.06
German     1          3.03
Dutch      1          3.03
Italian    1          3.03
Spanish    27         81.82
The input sentence is written in Spanish according to the trigrams.
Language   Score      %
English    0          0.00
French     0          0.00
German     0          0.00
Dutch      0          0.00
Italian    0          0.00
Spanish    5          100.00
Accuracy with bigrams: 100.00%
Accuracy with trigrams: 83.33%
Process time for a short sentance, using bigrams: 0.00 ms
Process time for a short sentance, using trigrams: 0.00 ms
Process time for a long sentance, using bigrams: 0.00 ms
Process time for a long sentance, using trigrams: 0.00 ms
PS D:\School\BD02\BLok_1_2023\PI7\PI7_inleveropdracht_3>
```

Figuur 1: Resultaat na het invoeren van een Spaanse zin.

6. Bibliografie

Great Learning Team. (2022, oktober 24). *An Introduction to Bag of Words (BoW) | What is Bag of Words?* Retrieved from mygreatlearning: https://www.mygreatlearning.com/blog/bag-of-words/

IBM. (n.d.). *What are recurrent neural networks?* Retrieved from ibm: https://www.ibm.com/topics/recurrent-neural-networks

Jurafsky, D., & Martin, J. H. (2023, januari 7). N-gram Language Models. Stanford, California, Amerika. Retrieved from https://web.stanford.edu/~jurafsky/slp3/3.pdf

Karani, D. (2018, September 1). *Introduction to Word Embedding and Word2Vec*. Retrieved from towardsdatascience: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

Mattingly, W. J. (2022). *Bigrams and Trigrams*. Retrieved from python-textbook: https://python-textbook.pythonhumanities.com/04_topic_modeling/04_01_04_bigrams.html#:~:text=Bigrams%20are%20two%20words%20that,distinct%20meaning%20when%20used%20together.