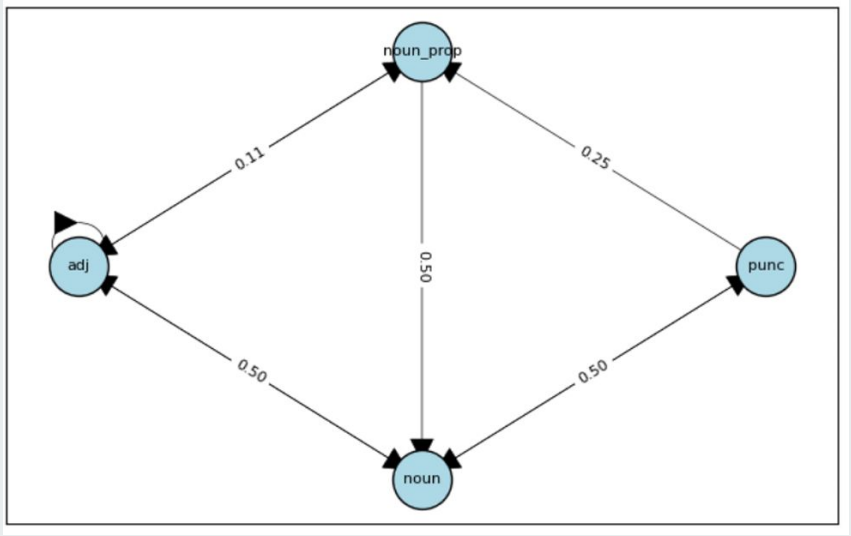
Arabic POS NetworkX



Building a Arabic POS Tagging Pipeline with Network Graphs

1. **Data Preprocessing**
   - Prepare the data for processing.
2. **Optimization**
   - Optimize the time for processing each word in the article.
3. **Techniques**
   - Use any suitable techniques to solve the problem.
4. **Deep Learning (Optional)**
   - Compare different deep learning methods and identify the best one.

Approaches

- **Rule-based POS tagging(lexicon,context rules(ال+اسم=صفة))**
- **Stochastic (Probabilistic) tagging**
- **Deep learning models.**

POS Tagging using Hidden Markov Models (HMM)

A **Hidden Markov Model (HMM)** is a statistical model that represents systems with hidden states. An HMM consists of:

- **States:** These represent the hidden states of the model. In the context of POS tagging, these are the POS tags themselves (e.g., noun, verb, adjective).
- **Observations:** These are the observed data, which in the case of POS tagging are the words in the sentence.
- **Transition Probabilities:** The probability of transitioning from one state to another.
- **Emission Probabilities:** The probability of a state emitting a particular observation.
- **Initial Probabilities:** The probabilities of the initial states.

Viterbi algorithm

To find the most likely sequence of POS tags for a given sequence of words, the **Viterbi Algorithm** is used. This **dynamic programming** algorithm finds the most probable sequence of hidden states (POS tags) given the observations (words).

- **Dynamic Programming (DP) Matrix (V)**: This matrix stores the probabilities of the most likely sequence of tags that end in each state (tag) at each time step.
- **Backpointer (BP) Matrix (B)**: This matrix stores the backpointers that record the path taken to achieve the highest probability at each step.

Data Set

I utilized a large corpus of **Arabic text** article scraped from the **Youm7** website. The corpus contains **341** sentences of Arabic text. Additionally, to further **test** our pipeline, we generated **5 new sentences using ChatGPT**, based on the content of these article.

- **Total Articles** : One article(contains 341 Arabic sentences) .
- **Total Sentences:** 341 sentences + 5 sentences for testing pipeline.
- **Total Tags:** 341 list of tags for each sentence (created using Camel Tool Pos Tagger).

CAMEL TOOLS POS TAGS

PoS tag	Description	Example (Arabic)	Example (English)	Description (Arabic)
abbrev	abbreviation	د.	Dr.	اختصار
adj	adjective	كبير	big	صفة
adv	adverb	بسرعة	quickly	ظرف
adv_interrog	interrogative adverb	كيف	how	ظرف استفهام
adv_rel	relative adverb	حيث	where	ظرف رابط
conj	conjunction	و	and	حرف عطف
conj_sub	subordinating conjunction	لأن	because	حرف عطف سببي
digit	digital numbers	١٢٣	123	أرقام رقمية
foreign	foreign	إنترنت	Internet	أجنبي
interj	interjection	آه	oh	حرف تعجب
noun	noun	كتاب	book	اسم
noun_prop	proper noun	أحمد	Ahmed	اسم علم
noun_quant	quantity noun	كثير	many	اسم كمية
part	particle	قد	might	حرف
part_det	demonstrative particle	هذا	this	حرف إشارة
part_focus	focus particle	حتى	even	حرف تركيز
part_fut	future marker particle	سوف	will	حرف مستقبل
part_interrog	interrogative particle	هل	do (interrogative)	حرف استفهام
part_neg	negative particle	لا	no/not	حرف نفي
part_verb	verbal particle	أن	that	حرف فعلي
part_voc	vocalized particle	يا	oh (vocative)	حرف نداء
prep	preposition	في	in	حرف جر
pron	pronoun	هو	he	ضمير
pron_dem	demonstrative pronoun	هذا	this	ضمير إشارة
pron_interrog	interrogative pronoun	من	who	ضمير استفهام
pron_rel	relative pronoun	الذي	which	ضمير موصول
punc	punctuation	،	,	ترقيم
verb	verb	كتب	wrote	فعل
verb_pseudo	pseudo verb	عسى	may (pseudo verb)	فعل شبه كامل
xxx	other	—	—	آخر

Preprocessing(Cleaning Articles)

- **Remove Tashkeel (Diacritics)** : Stripping diacritical marks to simplify the text.
- **Normalize Alef** : Standardizing different forms of the letter Alef.
- **Normalize Hamza:** Unifying the representation of Hamza.
- **Normalize Ya:** Standardizing different forms of the letter Ya.
- **Normalize Ta Marbuta:** Converting Ta Marbuta to its standard form.
- **Remove Emojis:** Eliminating any emojis present in the text.

How HMM works(Initialization)

1. **Define the number of tags (num_tags) and words (num_words).**
2. **Create the following matrices and vectors initialized to zero:**
   - A Matrix (num_tags x num_tags): Transition count matrix.
   - B Matrix (num_tags x num_words): Emission count matrix.
   - π Vector (num_tags): Initial probabilities vector.
3. **Define dictionaries to map tags and words to their indices:**
   - tag2idx: Maps each tag to a unique index.
   - word2idx: Maps each word to a unique index.

How HMM works(Populate Matrices)

For each sentence in the corpus:

- Extract the list of (word, tag) pairs.

Process Each Word-Tag Pair:

1. If the word is the first in the sentence:
    - Increment the initial probability count for the tag.
2. If the word is not the first in the sentence:
    - Update the transition count from the previous tag to the current tag.
3. Update the emission count for the current tag and word.

How HMM works(Normalize Matrices)

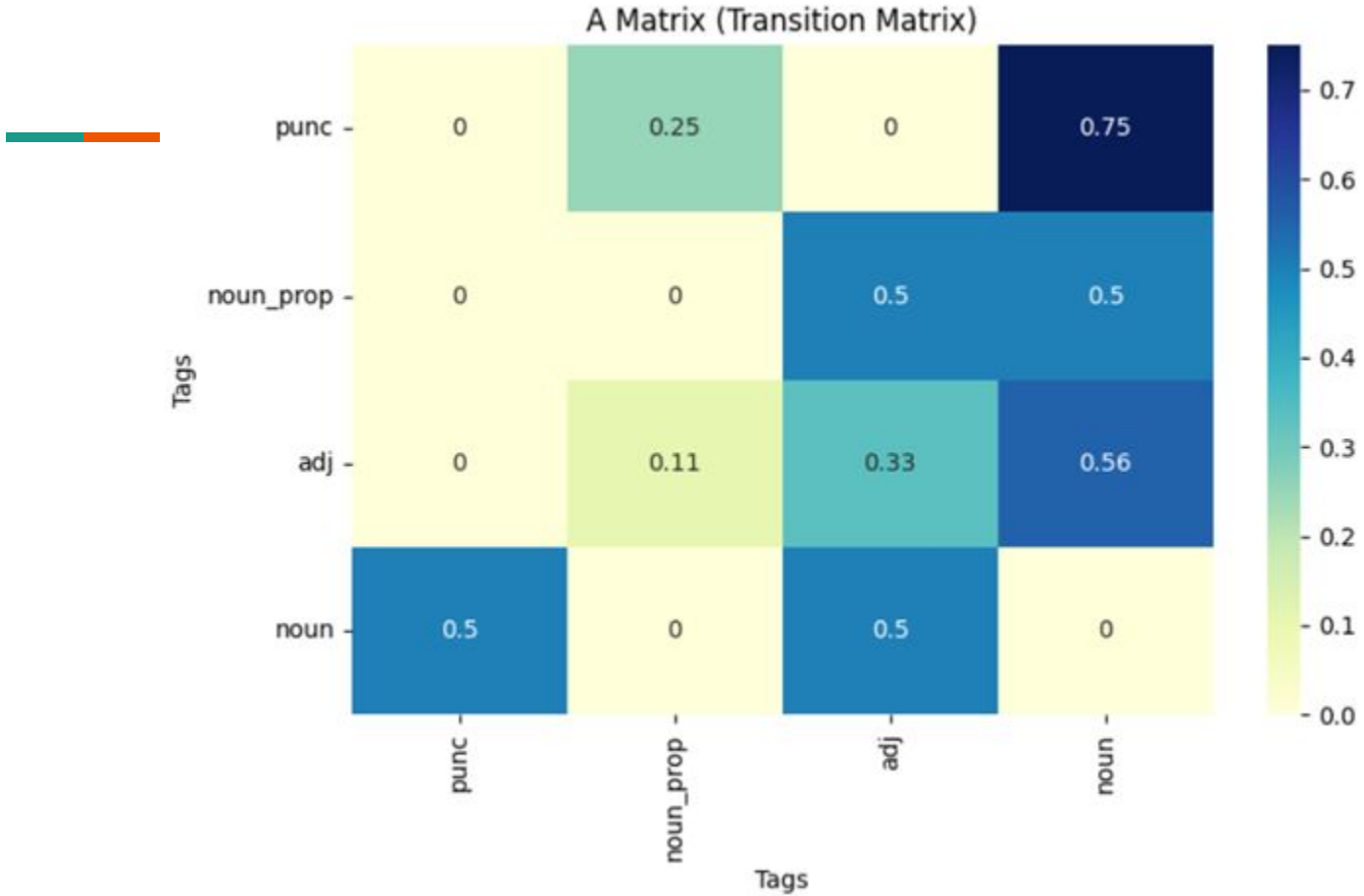
1. Normalize the A(**Transition Probabilities**) Matrix:
    - Convert transition counts to probabilities by dividing each row by its sum.
2. Normalize the B(**Emission Probabilities**) Matrix:
    - Convert emission counts to probabilities by dividing each row by its sum.
3. Normalize the π(**Initial Probabilities)** Matrix:
    - Convert counts to probabilities by dividing each row by its sum.

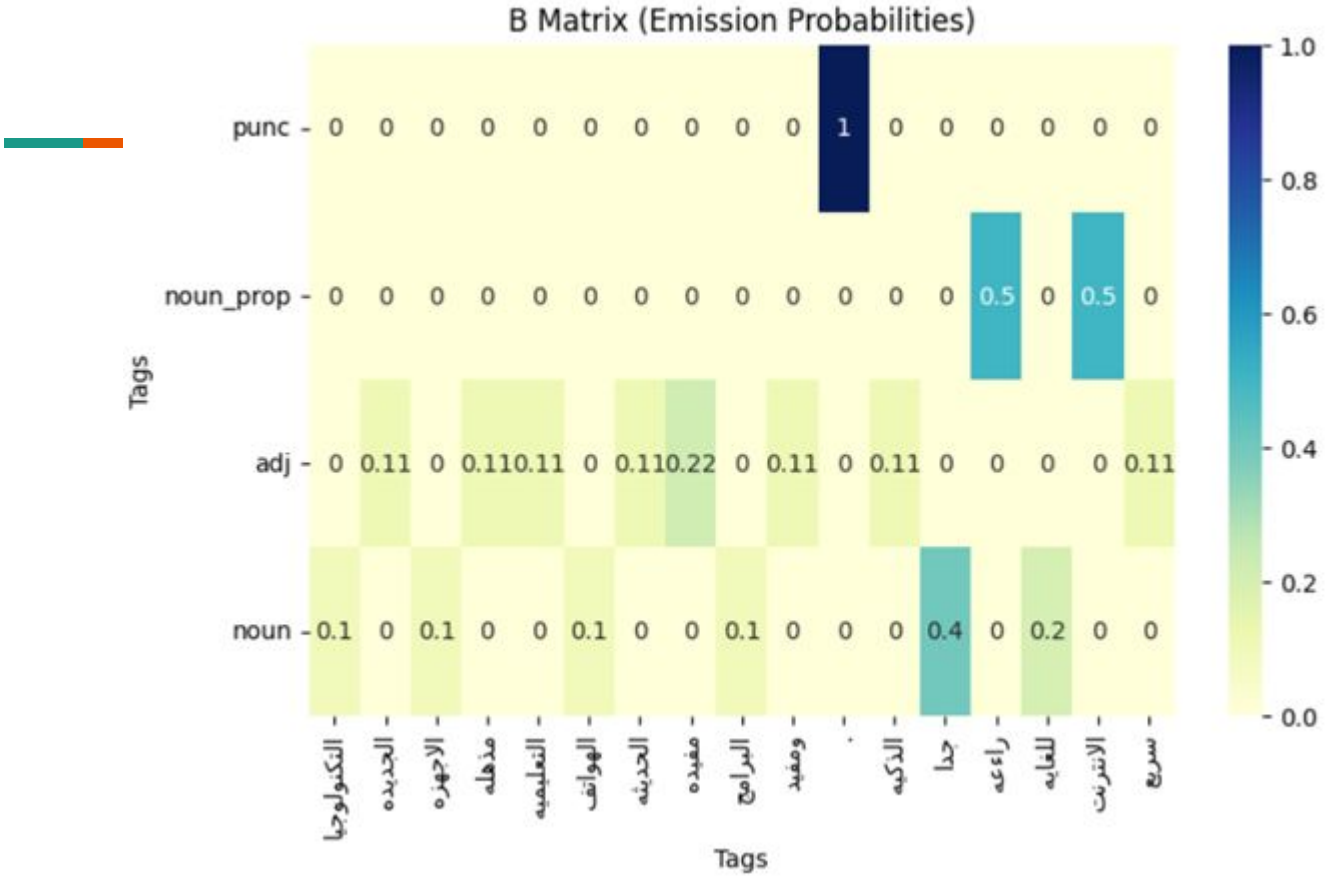
Example

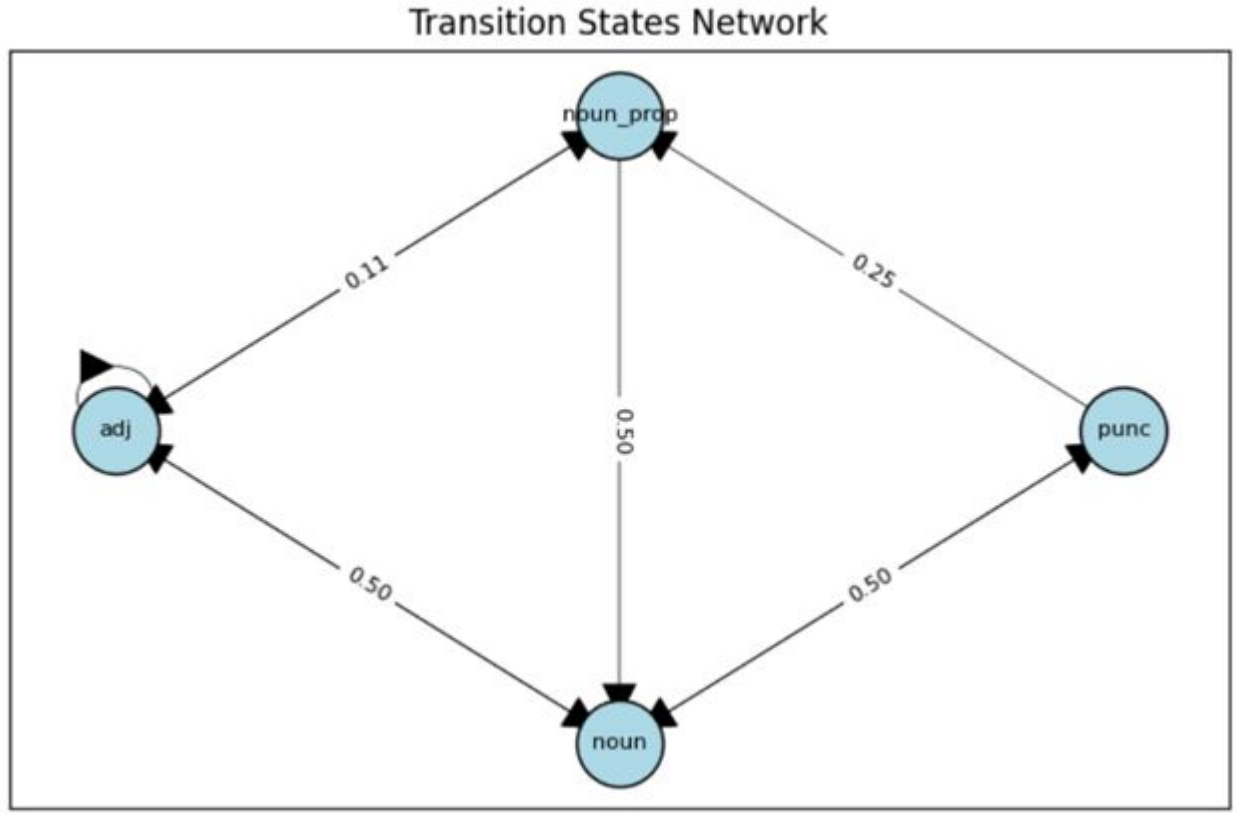
"التكنولوجيا الحديثة مفيدة جداً. الأجهزة الجديدة مذهلة للغاية. البرامج التعليمية رائعة جداً. الإنترنت سريع جداً ومفيد للغاية. الهواتف الذكية مفيدة جداً."

Tags Founds :

- Noun.
- Adj.
- Noun_prop.
- punc.


A Matrix (Transition Matrix)
punc
noun_prop
adj
noun
Tags
0
0.25
0
0.75
0
0
0.5
0.5
0
0.11
0.33
0.56
0.5
0
0.5
0
punc
noun_prop
adj
noun
Tags
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0


B Matrix (Emission Probabilities)
Tags
punc
noun_prop
adj
noun
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0.5 0 0.5 0
0 0.11 0 0.11 0.11 0 0.11 0.22 0 0.11 0 0.11 0 0 0 0 0.11
0.1 0 0.1 0 0 0.1 0 0 0.1 0 0 0 0.4 0 0.2 0 0
التكنولوجيا
الجديده
الاجهزه
مذهله
التعليميه
الهواتف
الحديثه
مفيده
البرامج
ومفيد
.
الذكيه
جدا
راءعه
للغايه
الانترنت
سريع
Tags
1.0
0.8
0.6
0.4
0.2
0.0


Transition States Network
noun_prop
adj
punc
noun
0.11
0.25
0.50
0.50
0.50

How Viterbi Algorithm Works(Initialization)

- Create a DP matrix (**dp**) and a BP matrix (**bp**).
- Initialize the first column of **dp** with the initial probabilities multiplied by the emission probabilities of the first observation.

How Viterbi Algorithm Works(Recursion)

- For each time step t from 1 to T−1:
  - For each state s:
    - Update dp[s, t] with the maximum probability of reaching state s at time t multiplied by the emission probability of the current observation.
    - Record the state that gave the maximum probability in **bp[s, t].**

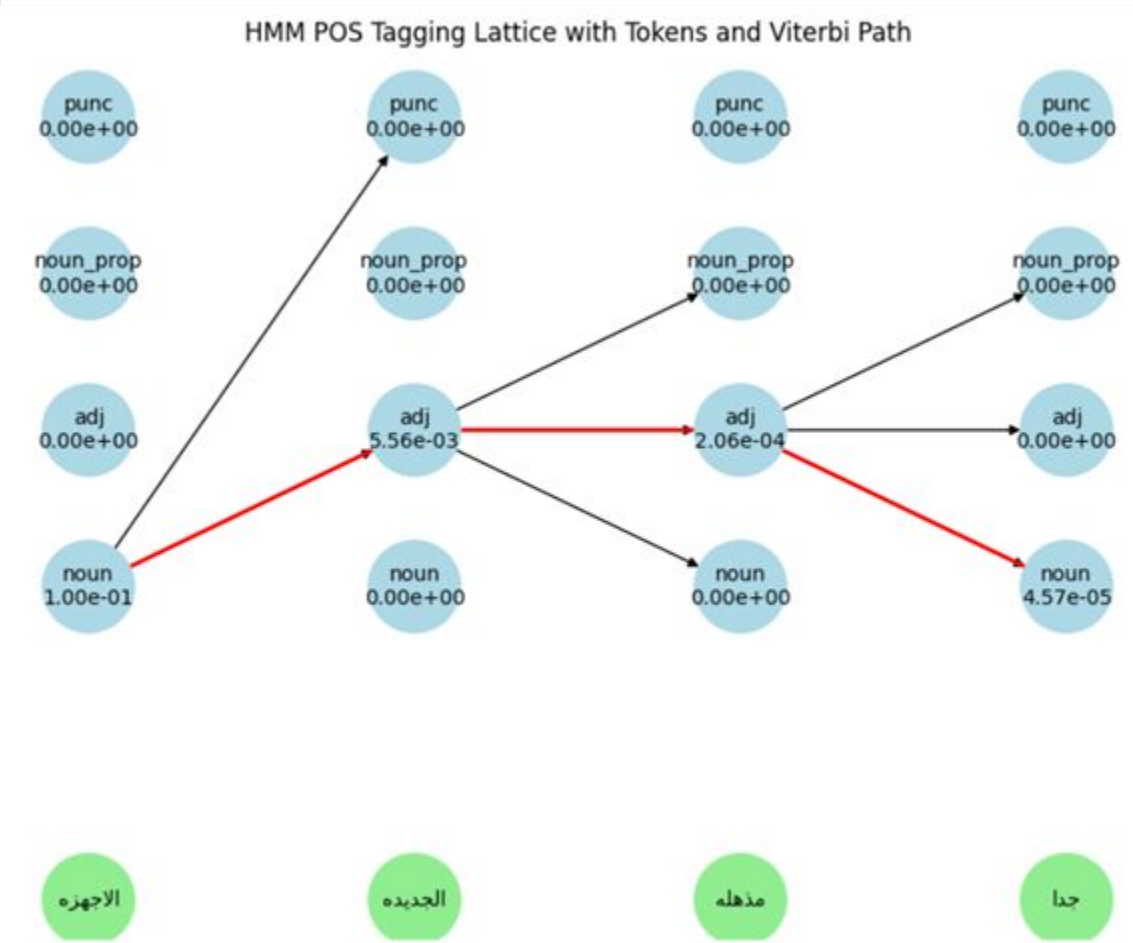
How Viterbi Algorithm Works(Termination)

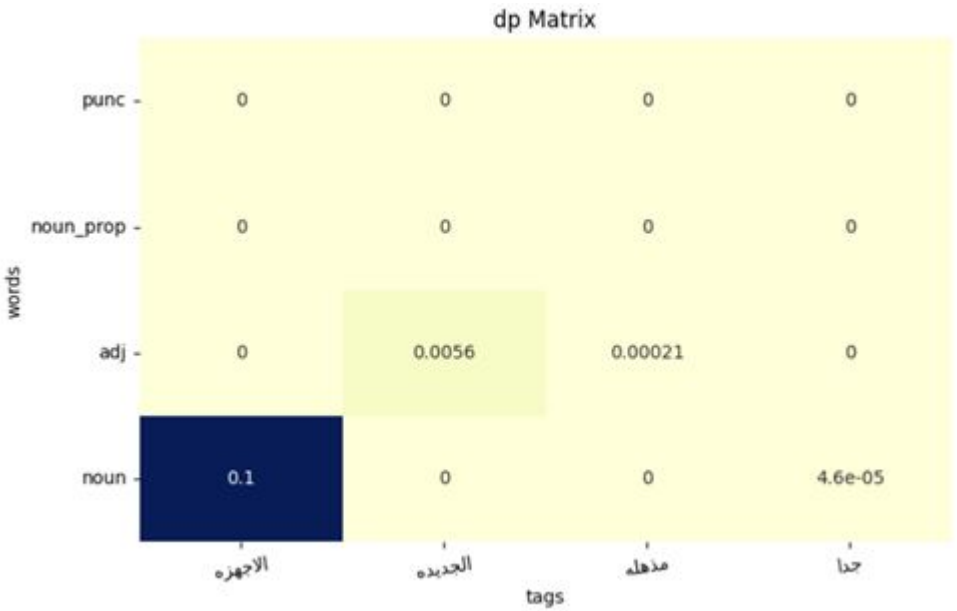
- Identify the state with the highest probability in the last column of **dp**.
- Backtrack through **bp** to find the most likely sequence of states (POS tags).

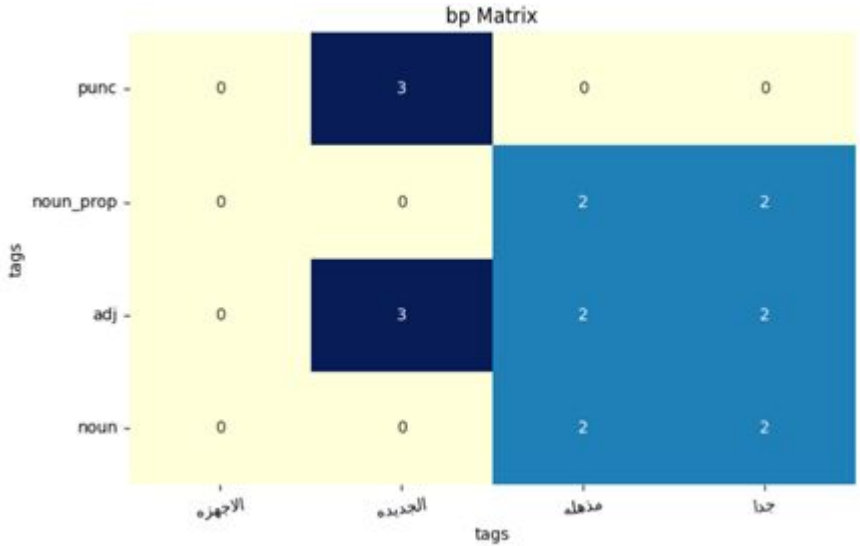
Why we use Viterbi Algorithm ??

Computational Cost:

- **Naive Approach:**
  - Number of possible sequences: $\mathbf{S^T}$
  - For example: $4^6 = 4096$ sequences
- **Viterbi Algorithm:**
  - Computational cost: $\mathbf{O(S^2 \cdot T)}$
  - For example: $4^2 \cdot 6 = 96$ operations


HMM POS Tagging Lattice with Tokens and Viterbi Path
punc
0.00e+00
punc
0.00e+00
punc
0.00e+00
punc
0.00e+00
noun_prop
0.00e+00
noun_prop
0.00e+00
noun_prop
0.00e+00
noun_prop
0.00e+00
adj
0.00e+00
adj
5.56e-03
adj
2.06e-04
adj
0.00e+00
noun
1.00e-01
noun
0.00e+00
noun
0.00e+00
noun
4.57e-05
الاجهزه
الجديده
مذهله
جدا


dp Matrix
punc
noun_prop
adj
noun
words
0
0
0
0
0
0
0
0
0
0.0056
0.00021
0
0.1
0
0
4.6e-05
الاجهزه
الجديده
مذهله
جدا
tags


bp Matrix
punc
noun_prop
adj
noun
tags
0
3
0
0
0
0
2
2
0
3
2
2
0
0
2
2
الاجهزه
الجديده
مذهله
جدا
tags

POS Tagging using Deep Learning models

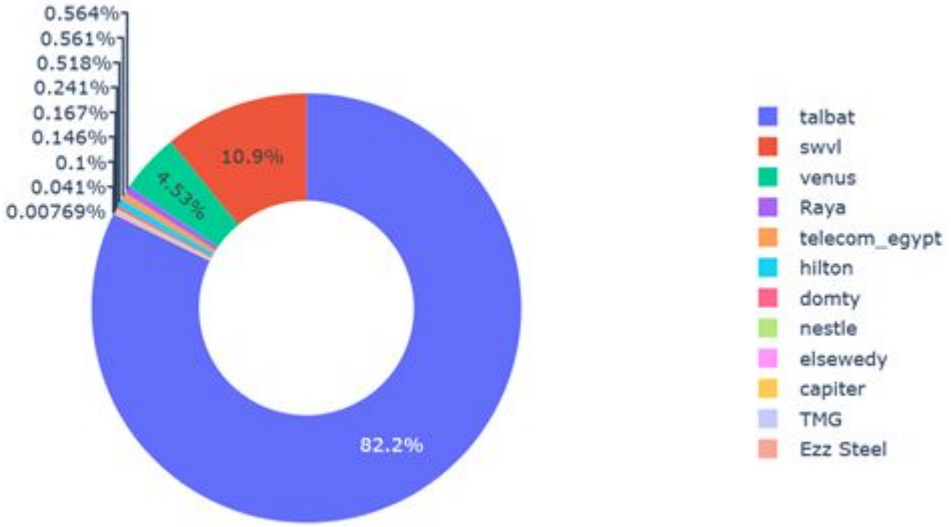
- Bi-LSTM (Bidirectional LSTM)
    a. **Description**: Processes sequences in both forward and backward directions.
    b. **Components**: Input gate, Forget gate, Output gate, Cell state.
    c. **Complexity**: Higher due to multiple gate calculations.
    d. **Pros**: Better at capturing long-term dependencies.
    e. **Cons**: More computationally intensive, longer training times.
- Bi-GRU (Bidirectional GRU)
    a. **Description**: Processes sequences bidirectionally like Bi-LSTM.
    b. **Components**: Update gate, Reset gate.
    c. **Complexity**: Lower with fewer gate calculations.
    d. **Pros**: More efficient, faster training, less prone to overfitting.
    e. **Cons**: May not capture long-term dependencies as effectively as LSTM.

Data Set

The data was collected for the purpose of sentiment analysis in order to produce a score for a given company. The data has 40K+ reviews in Arabic for sentiment analysis each labelled with a rating and its associated company name.

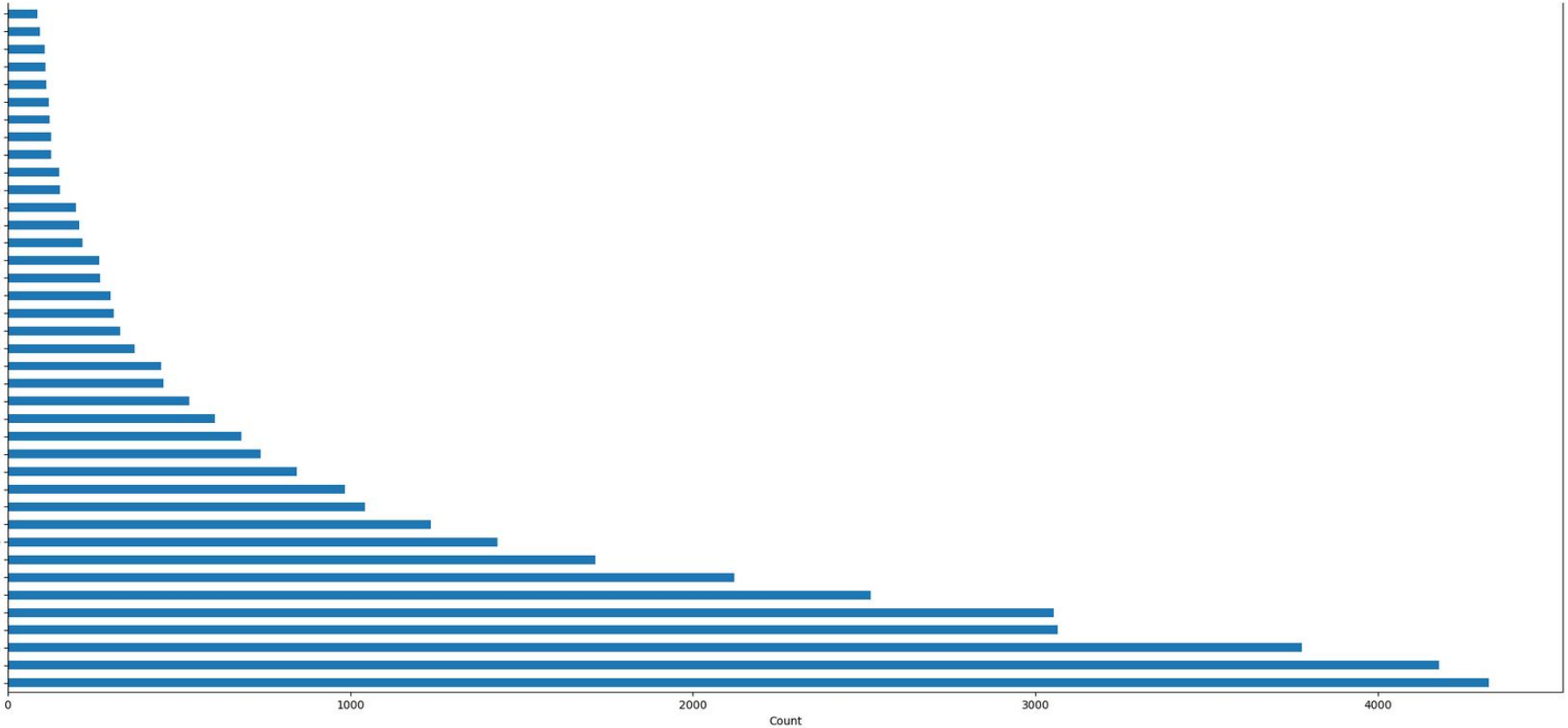
- **Training Set Size**: 30,733 sentences(<=60 tokens).
- **Validation Set Size**: 5,762 sentences(<=60 tokens).
- **Test Set Size**: 1,922 sentences(<=60 tokens).

Companies Reviews counts in Data sets



Data Preprocessing(Cleaning)

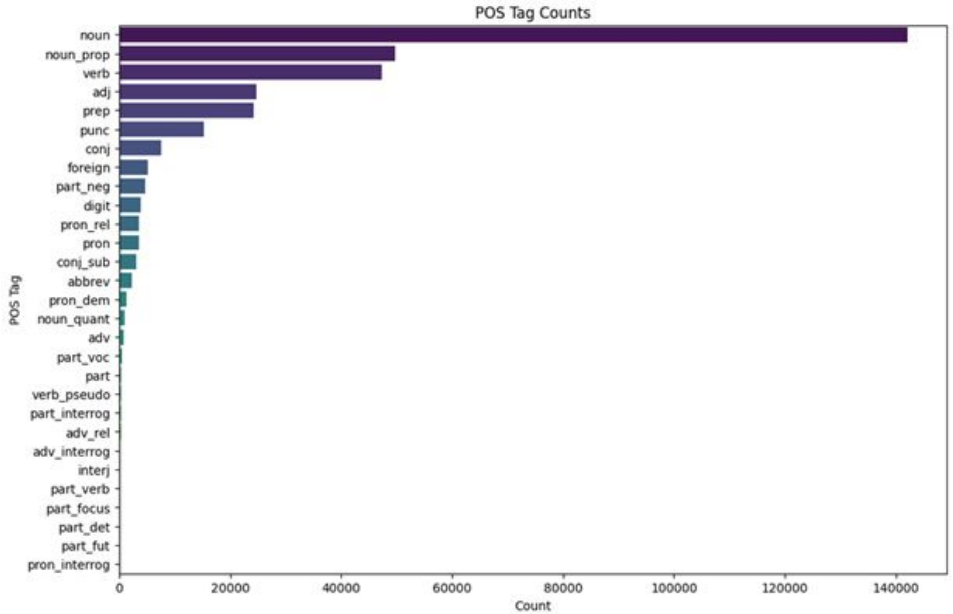
- **Data Cleaning**: Removed 1,042 duplicated reviews and checked for null reviews (none found).
- **Token Limit**: Removed samples with more than 60 tokens, as most examples contained between 2 and 60 tokens.
- **Distribution**: The graph shows the majority of examples fall within the 2 to 60 token range.


39
38
36
35
37
33
34
31
32
30
29
28
26
27
24
25
23
22
21
20
18
19
17
16
15
14
13
12
11
10
9
8
7
6
5
1
4
3
2
0
1000
2000
3000
4000
Count

Not Use Lemm or stemm

	nonLemmatizeToken	LemmatizeToken	nonLemmatizeTokenTag	LemmatizeTokenTag	correctLemm	stemToken	stemTokenTag	correctstemm
0	اسعارهم	اسعارهم	noun	noun	✔	اسعار	noun	✔
1	اغلا	اغلا	verb	verb	✔	اغل	verb	✔
2	من	من	prep	prep	✔	من	prep	✔
3	المحلات	محل	noun	noun	✔	محل	noun	✔
4	ب	ب	abbrev	abbrev	✔	ب	abbrev	✔
5	كثير	كثير	noun	noun	✔	كثير	noun	✔
6	و	و	conj	conj	✔	و	conj	✔
7	بحطولك	بحطولك	noun_prop	noun_prop	✔	حطول	noun_prop	✔
8	توصيل	توصيل	noun	noun	✔	توصيل	noun	✔
9	مجاني	مجان	adj	noun	✗	مجا	verb	✗
10	حكي	حك	verb	noun	✗	حك	noun	✗
11	فاضي	فاضة	adj	verb	✗	فاض	verb	✗
12	التطبيق	تطبيق	noun	noun	✔	تطبيق	noun	✔
13	لا	لا	part_neg	part_neg	✔	لا	part_neg	✔
14	انصح	انصاح	verb	noun_prop	✗	انصح	verb	✔
15	به	به	prep	prep	✔	به	prep	✔

Unbalanced POS Tags



Embedding Model: AraVec For the embeddings

I utilized the AraVec model, specifically the "**full_grams_cbow_300_twitter.mdl**" variant. AraVec is a pre-trained Arabic word embedding model that captures semantic information from a large corpus of Arabic text. The "**full_grams_cbow_300_twitter.mdl**" is trained using the **Continuous Bag of Words (CBOW)** approach with 300 dimensions on Twitter data.

Results

	modelName	Test-F1Score	InfernceTime(secs)
0	LSTM	0.929025	0.251356
1	BI-LSTM	0.937613	0.264170
2	GRU	0.929507	0.254330
3	BI-GRU	0.928979	0.259184
4	Ensembled	0.935815	0.599180