CS 613: NLP

Assignment 2: Tokenizer and Language Model Training for Malayalam Dataset

---

Heer Kubadia (22110096) | Jiya Desai (22110107) | Lavanya Lavanya (22110130) | Shrishti Mishra (22110246) | Utkarsh Srivastava (22110278)

17 November, 2024

Introduction

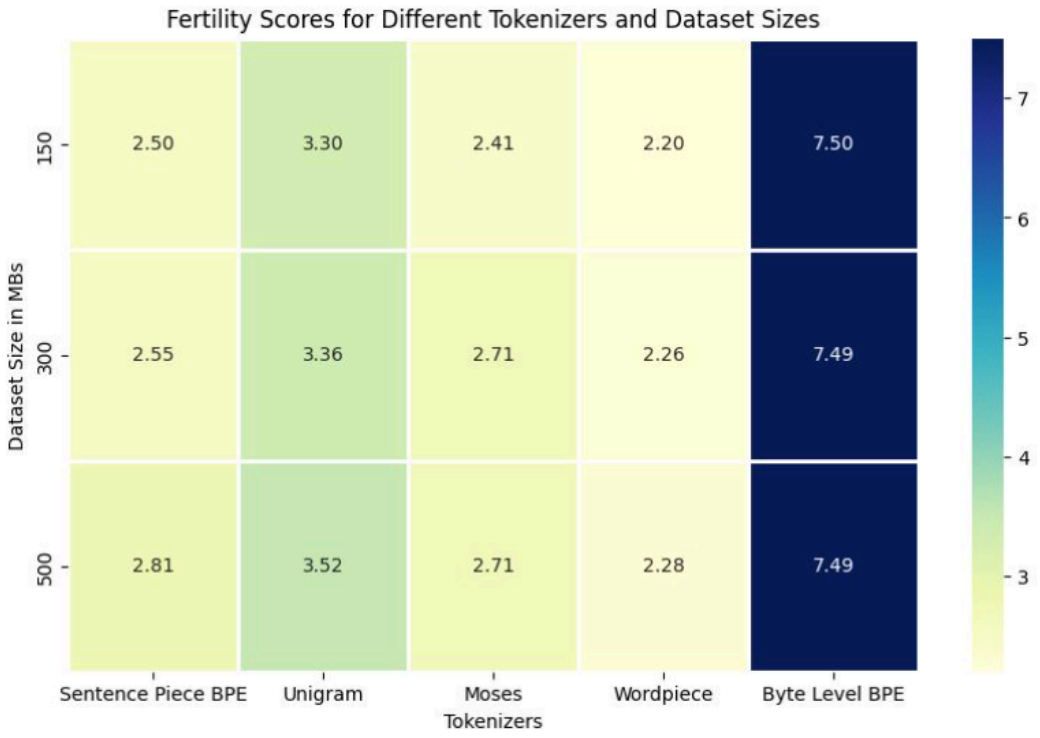
The objective of this project is to explore the effectiveness of different tokenizers trained on a custom dataset, evaluate their performance, and use the best tokenizer to train a language model with less than 100 million parameters.

Task 1: Tokenizer Training

We trained five distinct tokenizers, on randomly sampled datasets of varying sizes (150 MB, 300 MB, 500 MB), calculated the fertility score for each and selected the tokenizer with the least fertility score to tokenize our main dataset. The five tokenizers are:

1. Sentence Piece Byte Per Word (BPE) Tokenizer: This tokenizer uses Byte-Pair Encoding, a subword tokenization method that merges the most frequent pairs of bytes in the dataset iteratively. It creates tokens from byte-level subword units.
2. Unigram Tokenizer: The Unigram tokenizer uses a probabilistic model to split text into subwords by selecting tokens that minimize overall loss. Rather than merging pairs, it starts with a large pool of potential tokens and gradually removes the less likely ones.
3. Moses Tokenizer: The Moses tokenizer is a rule-based tokenizer primarily used for tokenizing text into words. It handles punctuation and whitespace tokenization according to language-specific rules
4. Wordpiece Tokenizer: The WordPiece tokenizer segments words into subwords based on a frequency-based algorithm. It splits rare or unknown words into smaller, more common subword units.
5. Byte Level Byte Per Word: This tokenizer is a variant of BPE that operates directly on byte sequences, rather than characters. Byte-level tokenization allows it to process any text, including special characters and Unicode, without requiring preprocessing.

The results for each tokenizer we tested are shown in the matrix given below:



Matrix showing the fertility scores for five different tokenizers on randomly sampled data of different sizes

As can be seen, the word piece tokenizer has the best fertility score for all data sizes. Also, wordpiece is highly effective for Malayalam due to its ability to handle the language's complex morphology, compound words, and inflectional nature. It reduces out-of-vocabulary issues by splitting rare or unknown words into meaningful subword units Hence, we used wordpiece to tokenize our main dataset for model training.

After selecting **wordpiece,** we tested it on five different datasets of approximately the same size, comprising scraped data. Among those five, we went ahead with Tokenizer-2 because it gave the lowest fertility score. The results have been shown in the table below:

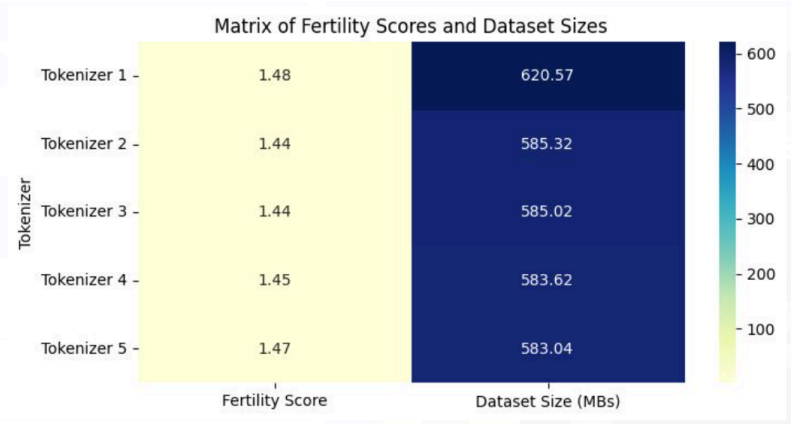


Table showing the performance of the wordpiece tokenizer on five different datasets

Task 2:

Once we had the tokenizer ready, the next step was to tokenize more and more data (that we had prepared in assignment 1 and use the tokenized data for training a model with a predefined architecture after making adjustments to the parameter size.

Model Selection and Training:

We wrote a Python script that trains and evaluates a language model on our original Malayalam dataset, implementing a smaller, resource-efficient version of the Llama model with lesser parameters. We first loaded our pre-trained Malayalam tokenizer (wordpiece, as decided in the first task), which was then used to tokenize the dataset into input_ids and attention_mask tensors for training.

The tokenized data is split into training and validation sets, and a custom MalayalamDataset class handles padding and length adjustments for uniformity. The model configuration uses LlamaConfig with reduced parameters: it has a hidden size of 768, only 8 attention heads, 4 key-value heads, and 12 hidden layers, all of which are scaled down from typical Llama configurations to make the total number of parameters lesser than 100 million, as required by the task. Training parameters are set with low max_steps and a small batch size to further optimize for resource constraints.

Results:

We recorded the model's perplexity at every 0.1 epoch, everytime we fed more data for training the model. As expected, the perplexity of the model kept decreasing as we used more training data. The following matrix tables show the result:

Initial training (600 MB):

Model Perplexity During Initial Training										
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Epoch 1	37509.75	37287.05	37160.11	37170.56	36950.37	36985.70	36925.71	36904.98	36981.47	36945.19
Epoch 2	36942.44	37047.05	36956.72	36891.39	36814.88	36790.52	36661.49	36704.00	36742.49	36773.93
Epoch 3	36681.78	36720.98	36786.45	36764.01	36843.09	36799.02	36780.80	36748.02	36721.64	36707.64
Epoch 4	36605.42	36669.19	36710.80	36719.61	36787.33	36782.10	36756.30	36765.38	36731.31	36717.34
Epoch 5	36800.59	36819.48	36844.98	37005.53	36965.53	37102.00	37072.15	37240.60	37274.46	37271.19
Epoch 6	37240.50	37089.44	36941.03	36909.17	36730.58	36769.00	36784.73	36751.53	36817.55	36923.53
Epoch 7	36851.24	36981.32	36829.21	36918.11	37021.50	37061.51	37191.13	37256.16	37256.41	37279.51
Epoch 8	37272.80	37276.53	37285.42	37292.07	37304.30	37300.25	37304.98	37303.09	37304.30	37304.33
Epoch 9	37305.76	37305.65	37306.79	37306.00	37307.15	37306.90	37307.20	37307.01	37307.15	37306.83
Epoch 10	37306.76	37306.47	37306.22	37306.01	37305.72	37305.40	37305.05	37304.84	37304.30	37304.00

Further training (feeding data in batches of 600 MBs):

Model Perplexity on Further Training										
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Epoch 1	23726.44	17622.89	15399.92	12844.47	10977.02	9480.92	8252.65	7469.63	6746.44	5954.26
Epoch 2	5468.86	5255.03	5100.75	4529.74	4151.70	3942.13	3559.05	3425.96	3267.03	2857.51
Epoch 3	2886.70	2844.32	2455.19	2270.98	2232.57	2094.32	2040.29	1846.04	1796.67	1660.97
Epoch 4	1521.41	1570.62	1448.36	1400.73	1371.92	1281.40	1268.88	1238.76	1159.42	1130.79
Epoch 5	1137.34	1061.75	1096.63	1047.07	1008.02	1040.95	1001.73	1018.72	1002.78	930.22
Epoch 6	962.35	975.95	923.95	897.59	932.44	861.79	895.61	859.23	845.24	835.57
Epoch 7	850.57	848.35	778.83	830.47	775.52	806.14	795.42	766.67	791.28	752.28
Epoch 8	766.87	722.72	755.59	727.85	752.59	698.40	738.08	720.13	687.01	712.14
Epoch 9	676.11	724.89	665.71	692.43	681.32	661.16	686.30	652.48	677.51	652.45
Epoch 10	671.55	647.39	650.12	665.64	631.46	656.90	635.86	662.18	642.74	643.26

Final trained model (after training on several GBs):

Final Model Perplexity										
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Epoch 1	561.67	474.77	453.83	421.00	422.99	423.14	406.76	428.72	414.48	423.37
Epoch 2	409.71	419.17	412.36	414.36	422.07	410.97	420.10	395.51	422.59	397.42
Epoch 3	404.57	405.94	403.69	394.44	412.70	394.84	406.77	396.10	406.43	388.53
Epoch 4	403.61	391.08	408.61	380.67	420.89	379.17	405.96	393.36	399.32	392.23
Epoch 5	398.25	387.30	401.85	391.56	394.21	393.01	405.75	385.76	395.03	402.37
Epoch 6	392.21	407.33	390.08	403.35	392.97	393.50	393.40	398.05	388.79	396.83
Epoch 7	387.97	391.09	389.66	393.23	386.04	397.04	389.62	390.96	389.11	382.94
Epoch 8	388.38	388.60	382.00	389.38	386.68	385.87	388.49	385.82	388.75	382.57
Epoch 9	387.40	385.58	387.32	383.36	387.74	380.41	390.71	384.19	386.39	388.85
Epoch 10	384.66	385.68	386.62	385.92	382.93	390.88	384.32	391.85	383.31	394.23

As can be seen, the final model perplexity came out to be 394.23.

Testing the model on simple prompts (such as capital of Kerala, food items in kerala, etc):

Prompt 1: കേരളത്തിന്റെ തലസ്ഥാനം **(Capital of Kerala)**

Output 1: കേരളത്തിന്റെ തലസ്ഥാനം വിദഗ്ധ അച്ചടക്ക അച്ചടക്ക സീസ സീസ ഇരുവിഭാഗ സെന്റര് സെന്റര് സെന്റര് സെന്റര് സെന്റര് പാരാലി സെന്റര് അനുമതി അനുമതി അനുമതി അനുമതി അനുമതി തുറക്കുന്ന അനുമതി തുറക്കുന്നദി തുറക്കുന്ന തുറക്കുന്ന തുറക്കുന്ന തുറക്കുന്ന തുറക്കുന്ന തുറക്കുന്ന തുറക്കുന്ന തുറക്കുന്ന തുറക്കുന്ന തുറക്കുന്ന തുറക്കുന്ന ഗവ തുറക്കുന്നരിച്ചത് തുറക്കുന്നരിച്ചത് തുറക്കുന്നരിച്ചത് തുറക്കുന്നരിച്ചത് തുറക്കുന്ന സെന്റര് സെന്റര് സെന്റര് സെന്റരിച്ചത്

Prompt 2: കേരളത്തിലെ പ്രശസ്ത പുഴകൾ **(Famous rivers in Kerala)**

Output 2: കേരളത്തിലെ പ്രശസ്ത പുഴകൾ ഹാ ഹാ കക്ഷി ഒന്ന കക്ഷിക്ക്ക്ക്ക്ക്ക്ക്ക്ക്ക്ക്ക്ക് പി jul kerala ുകൾ : jul ുകൾ jul ുകൾുകൾ jul ുകൾ jul ുകൾ jul ുകൾ jul ുകൾ jul

Prompt 3: കേരളത്തിലെ പ്രധാന ഭക്ഷണം **(Main food in Kerala)**

Output 3: കേരളത്തിലെ പ്രധാന ഭക്ഷണം ഡോവലിനും ജയിലിൽ ക്രിസ്റ്റ ക്രിസ്റ്റ വീട ചെയ്യ ചെയ്യ ചെയ്യ കോർപറേഷൻായി കോർപറേഷൻ കോർപറേഷൻൾഡോ left ബിൾഡോ left left left left left left left left left left leftതൊന്നും ഇപ്പോള്തൊന്നും ഇപ്പോള്

Prompt 4: കേരളത്തിലെ വിനോദസഞ്ചാര കേന്ദ്രങ്ങൾ **(Tourist destinations in Kerala)**

Output 4: കേരളത്തിലെ വിനോദസഞ്ചാര കേന്ദ്രങ്ങൾ deshabhimani സംഗീതം സംഗീതം സംഗീതം സംഗീതംക്ക് സംഗീതംക്ക് ബെംഗളൂരുവിലെ ബെംഗളൂരുവിലെ ബെംഗളൂരുവിലെ ബെംഗളൂരുവിലെ ആശുപത്രിയില് ബെംഗളൂരുവിലെ ആശുപത്രിയില് തീപിടിത്തം ബെംഗളൂരുവിലെ ആശുപത്രിയില് തീപിടിത്തം തീപിടിത്തം തീപിടിത്തം തീപിടിത്തം തീപിടിത്തം വെന്തുമരിച്ചു വെന്തുമരിച്ചു വെന്തുമരിച്ചു വെന്തുമരിച്ചു വെന്തുമരിച്ചു വെന്തുമരിച്ചു വെന്തുമരിച്ചു വെന്തുമരിച്ചു വെന്തുമരിച്ചു വെന്തുമരിച്ചു വെന്തുമരിച്ചു വെന്തുമരിച്ചു ലൈം വെന്തുമരിച്ചു ലൈം വെന്തുമരിച്ചു ലൈം വെന്തുമരിച്ചു ലൈം വെന്തുമരിച്ചു നൃത്തസംവിധായകൻ നൃത്തസംവിധായകൻ നൃത്തസംവിധായകൻ

Prompt 5: കേരളത്തിലെ പ്രസിദ്ധ എഴുത്തുകാർ **(Famous writers from Kerala)**

Output 5: കേരളത്തിലെ പ്രസിദ്ധ എഴുത്തുകാർ deshabhimani പേർക്ക് സംഭവത്തിൽ വില സംഗീതജ്ഞ സംഗീതജ്ഞ സംഗീതജ്ഞ സംഗീതജ്ഞ സംഗീതജ്ഞ സംഗീതജ്ഞ സംഗീതജ്ഞ സംഗീതജ്ഞ എന്നാണ് & എന്നാണ് & എന്നാണ് &

Prompt 6: കേരളത്തിലെ പ്രധാന ആഘോഷങ്ങൾ **(Major festivals in Kerala)**

Output 6: കേരളത്തിലെ പ്രധാന ആഘോഷങ്ങൾ ഡോവലിനുംക്ക് ബാങ്ക് പ്രതി വേണ്ടത്ര ജയിലിൽ വേണ്ടത്ര വേണ്ടത്ര വേണ്ടത്ര വേണ്ടത്ര അസ് വേണ്ടത്ര അവരെ അസ് lead അസ് ഒരു lead അവരെ അസ് lead ഘട്ടം ഘട്ടം ഘട്ടം ഘട്ടംട്ട് ഒരുട്ട് ഒരു തുറക്കുന്നട്ട് ഒരുഗൽട്ട് ഒരുഗൽ കാലാവസ്ഥട്ട് കോട്ടയം വാല ഘട്ടം കാലാവസ്ഥട്ട് കോട്ടയം വാലുപ കോട്ടയം

Prompt 7: കേരളം ഭാഷ **(Language of Kerala)**
Output 7: കേരളം ഭാഷകൾ ശാസ്ത്രജ്ഞ (കുടുംബ ശാസ്ത്രജ്ഞ ശാസ്ത്രജ്ഞ (മുഖ്യമന്ത്രി ചികിത്സ (ചികിത്സ (്ത, (്ത, (്ത, (്ത,്ത, (

Prompt 8: ഇന്ത്യയുടെ തലസ്ഥാനം **(India’s capital)**
Output 8: ഇന്ത്യയുടെ തലസ്ഥാനം കഴിഞ്ഞ എന്ത് എന്ത് എന്ത് എന്ത് എന്ത് & പുറത്തുവിട്ട & പുറത്തുവിട്ട sports sports sports എന്ത് എന്ത് & & & in എന്ത് & in & in deshabhimani & in deshabhimani & deshabhimani & deshabhimani & deshabhimani & deshabhimani & ചെയ്യ in തൃശ്ശൂര് & തൃശ്ശൂര് & തൃശ്ശൂര് തൃശ്ശൂര് തൃശ്ശൂര് തൃശ്ശൂര് തൃശ്ശൂര്

Prompt 9: കേരളത്തിലെ കാഴ്ചകൾ
Output 9: കേരളത്തിലെ കാഴ്ചകൾ അഴിമതിയ ഹാ മേഖലകൾ ഹാ ഡിവൈ ഡിവൈതിരെതിരെതിരെ ഉരു ഉരു ഹാ ഹാ ഹാ ഹാ ഹാ ഹാ ഹാ്്്്്്്്്്്്്്്്്്് ശാസ്ത്രജ്ഞ ശാസ്ത്രജ്ഞ ശാസ്ത്രജ്ഞ ശാസ്ത്രജ്ഞ ശാസ്ത്രജ്ഞ ശാസ്ത്രജ്ഞ ശാസ്ത്രജ്ഞ ശാസ്ത്രജ്ഞ ശാസ്ത്രജ്ഞ ശാസ്ത്രജ്ഞ ശാസ്ത്രജ്ഞ thursday പരിധിയിൽ

Prompt 10: കേരളത്തിലെ തീരങ്ങൾ
Output 10: കേരളത്തിലെ തീരങ്ങൾ deshabhimaniിക്കുന്നിക്കുന്നിക്കുന്ന ((((സിനിമ സിനിമ സിനിമ,,,, malayalam news, malayalam news, malayalam news, malayalam daily, malayalam newspaper, daily newspaper, daily newspaper of kerala, lead news, lead news, world / international

Individual Contributions

For the first task, everyone tested one tokenizing technique each, and collaborated for the work further.

Heer Kubadia: Led the team in understanding the workflow of the entire process, tested the BPE tokenizer, compiled the tokenizer results provided by all the members and created the final tokenizer, wrote code for model training.
Jiya Desai: Tested the wordpiece tokenizer, organised the data into small batches for tokenization, wrote code for tokenization, ran code for training the model, created the report.
Lavanya Sharma: Tested the Moses tokenizer, organised the data into small batches for tokenization, wrote code for tokenization, ran code for tokenizing the data, helped in writing code for model training.
Shrishti Mishra: Tested the unigram tokenizer, identified the model best suited for the task, made adjustments to the parameter size, wrote code for model training, ran code for tokenizing the data.
Utkarsh Srivastava: Tested the byte level BPE tokenizer, identified the model best suited for the task, made adjustments to the parameter size, wrote and ran code for model training.