Gebze Technical University

Department of Computer Engineering

CSE 654 / 484

Fall 2024

Homework 1 – Report

1) Introduction

N-gram language models are statistical models that predict the next item in a sequence, such as the next character or syllable, based on the previous *n−1 n-1 n−1* items. In this project, I created both character-based and syllable-based N-gram models to understand which approach is better at language generation and lower perplexity for Turkish. The project involves processing the Turkish Wikipedia Dump dataset, normalizing and syllabifying the data, building 1-gram, 2-gram, and 3-gram models for both characters and syllables, and applying Good-Turing smoothing to handle unseen data. The objectives are to calculate perplexity, generate random sentences, and evaluate model suitability for Turkish.

2) Design and Implementation

Code Design:

-Load large datasets in chunks, reducing memory usage. I've deleted Chinese and Russian words from the dataset also I've deleted between '<' and '>' to get more accurate results.

-Normalize and syllabify text to prepare it for character and syllable-based N-gram modelling. In syllabify part, I've used an existing tool which is linked in the references part.

-Build and smooth N-gram models with Good-Turing smoothing for both character-based and syllable-based approaches.

-Calculate perplexity as an indicator of model performance.

-Generate random sentences based on the N-gram probabilities for qualitative evaluation.

Function Descriptions:

-) load_file_in_chunks(filename, chunk_size=1024)

Inputs: filename (file path), chunk_size (size of each data chunk in bytes).

Outputs: Yields chunks of data for each iteration.

Functionality: Reads a large file in small chunks to prevent memory overload.

-) normalize_text(text)

Inputs: text (raw text data).

Outputs: Normalized text (Turkish characters are converted to their English equivalents).

Functionality: Converts Turkish-specific characters to English and removes HTML tags and non-Latin characters.

-) syllabify_text(text)

Inputs: text (normalized text).

Outputs: Syllabified text (text tokenized into syllables).

Functionality: Uses the Encoder class to tokenize text into syllables and removes non-Latin characters.

-) build_ngram_dict(data, n)

Inputs: data (text data), n (N-gram size).

Outputs: Dictionary of N-grams and their frequencies.

Functionality: Generates an N-gram dictionary for character-based models.

-) build_syllable_ngram_dict(data, n)

Inputs: data (syllabified text), n (N-gram size).

Outputs: Dictionary of syllable N-grams and their frequencies.

Functionality: Generates an N-gram dictionary for syllable-based models.

-) good_turing_smoothing(n_gram_dict)

Inputs: n_gram_dict (N-gram frequency dictionary).

Outputs: smoothed_prob_dict (dictionary of smoothed probabilities) and unseen_prob (probability for unseen N-grams).

Functionality: Applies Good-Turing smoothing by adjusting the counts of N-grams based on frequency.

-) calculate_perplexity(test_data, n, smoothed_prob_dict, unseen_prob)

Inputs: test_data (test dataset), n (N-gram size), smoothed_prob_dict (smoothed probabilities), unseen_prob (unseen probability).

Outputs: Perplexity score.

Functionality: Calculates the perplexity of the test data based on N-gram probabilities.

-) get_top_5_ngrams(ngram_dict, prefix)

Inputs: ngram_dict (N-gram dictionary), prefix (N-1 gram).

Outputs: Top 5 N-grams and their probabilities.

Functionality: Filters and sorts top 5 most probable N-grams matching the prefix.

-) generate_sentence(ngram_dict, n, length=50, syllable_based=False)

Inputs: ngram_dict (N-gram dictionary), n (N-gram size), length (sentence length), syllable_based (flag for syllable model).

Outputs: Generated sentence.

Functionality: Generates a sentence based on N-gram probabilities, either character-based or syllable-based.

3) Results and Tables

N-Gram Size	Generated Sentence by Character Based Model	Generated Sentence by Syllable Based Model
1-Gram	e.ovamwrmias b ikeora,dr\nslklalo iecna a t atnyr,nb	yö-nın-ku-a-a-le-les-ta-dün-0-lat-le-bir-fil-sik-nü-de-ler-da-mı-su-ğu-de-sim-gib-lar-ça-bi-müş-ır-nu-,-man-ve-5-rin-rek-tır-sin-bir-sı-ti-vi-bir-bi-po-ri-lan-dır-o
2-Gram	ɲwlɲ alulu kir. Klastalinilin olinan bunandunil ki	kıl-dı-ğı-i-le-ri-nin-gü-ne-de-,1-9-6-yı-lın-da-ha-ne-ka-ra-da-ha-ne-ka-ra-sın-da-ki-li-bir-a-na-da-ya-da-ha-son-ra-fın-dan-1-5-0-1-9-9-9-8-6
3-Gram	iecinin icini ver koyna bat alamanlan ara karin bu	lı-,-a-rap-ça-'-nın-ba-ba-sı-dır-.-u-lus-la-ra-ve-ril-miş-tir-.-ma-hal-le-ye-rek-a-na-do-lu-'-nda-ki-,-a-ra-lık-1-9-7-1-yı-lın-da-a-dı-na-ya-rak-a

Model Type	1-Gram Preplexity	2-Gram Preplexity	3-Gram Preplexity
Char Based	23.032	284.43	2532.64
Syllable Based	2435.571	5727.27	24.81

N-Gram Size	Generated Sentence by Character Based Model	Generated Sentence by Syllable Based Model
1-Gram	ysi "nb r(gakmiaiek saa .cedz iara epr oaidrzar v	lin-sal-ri-kay-rın-de-8-vit-yün-,-çuk-le-tom-par-,-le-ti-nı-se-neyt-le-p-physi-lil-mun-na-kı-0-2-ğı-ter-yas-de-ta-be-yo-la-hin-sur-şa-ta-6-hi-la-km-çal-.-.-kan-ri
2-Gram	\x93dane stasirandoyeri kulalelile si klarisar a bole	jı-nı-la-rın-da-ha-re-ti-mi-ne-si-'de-,-1-9-9-8-8-0-1-2-0-0-0-0-.-1-9-5-'-in-gil-te-le-ri-ne-si-ne-de-e-ği-ni-den-ya-'-da-,-1-9
3-Gram	.(1999200 tan ilinin sozumusti. Anla da dun allami	cı-,-o-yun-cu-nun-da-gö-rev-yap-mak-tay-dı.-bu-ra-da-bu-lu-nur-.-s-0-ti-pi-bir-mer-cek-li-tip-bir-te-les-kop-la-keş-fe-dil-miş-tir-.-1-7-7-7-)-,-türk-si-ne

4) Analysis and Conclusion

Character-Based Model Analysis:

The character-based model showed a notable increase in perplexity as N increased from 1 to 3. While the 1-gram model had the lowest perplexity at 23.032, this figure rose significantly to 284.43 for the 2-gram model and even more so to 2532.64 for the 3-gram model. This trend implies that higher-order N-grams may not provide significantly better predictive accuracy for the character-based model, possibly due to sparsity issues or insufficient training data for capturing frequent trigrams.

Syllable-Based Model Analysis:

The syllable-based model perplexities exhibit an unexpected pattern. The 1-gram and 2-gram models have notably high perplexity values (2435.571 and 5727.27, respectively), indicating considerable uncertainty in predicting syllables independently and in pairs. However, the perplexity for the 3-gram model drops sharply to 24.81, far lower than even the character-based 1-gram model. This finding is counterintuitive, as syllables are generally more complex units than characters and would typically yield higher perplexity values. Nonetheless, the lower perplexity here suggests that the 3-gram syllable model has captured predictable patterns effectively. Also, the syllable method that is used in the project divides syllables with spaces so the syllable based model cannot provide spaces when generating the result. That's why it creates much more accurate sentences and words comparing with the character based model.

Overall, these results align with the observation that, as N increases, the accuracy of the model also improves, which is reflected in the more accurate (lower) perplexity scores for higher N-grams, especially evident in the syllable-based 3-gram model. The generated sentences align with these findings. As N increases, sentence quality improves, suggesting that higher-order N-grams provide more context, enhancing the quality and coherence of generated sentences. Despite some inconsistencies in perplexity trends, this improvement aligns with expectations that more context results in more fluent predictions.

Conclusion

Higher N-gram values generally enhance predictive accuracy, particularly for syllable-based models, with the syllable-based 3-gram model outperforming other configurations in terms of perplexity.

This suggests that syllable-based trigrams are effective for modelling Turkish, likely due to their ability to capture nuanced patterns in Turkish syllabic structures.

5) Table on LLM Usage and Resources

Qodo-Gen and Chat-Gpt is used as LLM's in this project.

Code Design and Implementation	**Report**
Enhanced already written functions such as chunk loading the dataset using chunks method is an advice from chat gpt	To be able to write more formal and well designed sentences
Commenting the functions.	
To be able to write the code much faster such as getting simple structure of a method.	
Getting help especially in calculating the preplexity and good turing smoothing methods' math part.	

Additional Resources:

https://github.com/ftkurt/python-syllable

Used to syllable Turkish words.

https://www.kaggle.com/datasets/mustfkeskin/turkish-wikipedia-dump

Main dataset that is used in the project.