Project II: Modeling language

Language as Data at Göttingen University

Moujan Mirjalili, Farhan Kayhan, Merle (Joris) Hellwig 19.12.2024

Deliverable 1

Pre-Processing for Persian

1. Overview of Pre-Processing Steps

For the Persian dataset, the preprocessing pipeline was designed to normalize, clean, and tokenize text effectively, considering the unique characteristics of the Persian language. The following steps were implemented using the Hazm library for Persian text processing, along with Python standard libraries.

2. Pre-Processing Pipeline

- Normalization: Hazm's Normalizer was applied to standardize the text. Hazm normalization converts text to a standard form, such as removing diacritics, correcting spacing, etc. for example:

  اصلاح نویسه ها و استفاده از نیم‌فاصله پردازش را آسان می کند ← اصلاح نویسه‌ها و استفاده از نیم‌فاصله پردازش را آسان می‌کند

- Removing Unwanted Characters: A custom list of Persian and non-Persian punctuation marks (punc = '''()-[]{};،:'"\\, <>./?@#$%^&*_~.''') was defined and removed using regular expressions. This ensured that punctuation symbols did not interfere with tokenization. Common Persian-specific invisible characters such as zero-width non-joiners (\u200c), directional markers (\u200e, \u200f), and line breaks (\n) were replaced with spaces.

- **Stemming (Optional)**

  The Hazm Stemmer was available for stemming words. However, stemming was not applied in this setup to preserve the full vocabulary for training the language models, as it was stemming too much of the words for our purpose. For example: کتاب‌ها → کتاب

3. Tokenization

The normalized text was tokenized into words using Hazm's word_tokenize() function, which is tailored for Persian's grammatical and syntactic rules. Tokenization split the sentences into lists of words, preserving the integrity of Persian words.

4. Tools and Libraries Used

- Python version: 3.10.12
- Hazm library version: Latest stable version as of December 2024 (Normalizer, Stemmer, word_tokenize)

This preprocessing pipeline effectively prepares Persian text for language modeling tasks by addressing challenges such as inconsistent writing styles, handling unique characters, and tokenizing correctly. However, Persian’s rich morphology and variations in word forms (e.g., prefixes/suffixes) remain a challenge for pre-processing.

Pre-Processing for Spanish

1. Overview of Pre-Processing Steps

For the Spanish dataset, the preprocessing pipeline was designed to clean and normalize the text, making it suitable for further analysis or modeling tasks. The following steps were applied using Python's standard libraries, with specific emphasis on handling typical Spanish characters and formatting. The approach was kept basic in this exercise, with some variations in later exercise.

2. Pre-Processing Pipeline

- The leading numbers at the beginning of each line were stripped to remove any irrelevant numeric identifiers. The text was split using the `.split("\t")` method. This command split the data based on tab characters (`\t`). Since the dataset did not contain any tab characters, the split method was applied to separate the lines into two parts. The second part of each split was used for the regular expression replacement process.
- Removing Non-Typical Characters: A regular expression (Regex) was used to remove all characters that were not typical in the Spanish language. This Regex pattern, `[^0-9a-zA-Z\sáéíóúüñ]`, ensured that only alphanumeric characters, whitespace, and common Spanish diacritics (like accents and tilde) were retained.
- Regular Expression for Replacement: The Python `re` module was used for replacing unwanted characters. The regular expression was executed with the Python `re` library, version "2024.11.6".

3. Tokenization

The cleaned text was split into individual words or tokens using a basic tokenization method. This ensured that the text was separated into manageable units for further processing. The used Tokenizier was gpt2 tokenizier from the tiktoken python Module with the version 0.8.0.

4. Tools and Libraries Used

- Python version: 3.10.14
- Other libraries: Python's standard libraries were used for string manipulation, file handling, and splitting.

This preprocessing pipeline ensures that the Spanish text is cleaned and tokenized appropriately for further analysis, ensuring that non-relevant or non-Spanish characters are excluded. We measure the performance of this preprocessing step in Deliverable 4 against other Methods

Deliverable 2

N-Gram Baseline

1. N-Gram Model Description

The n-gram model is a statistical language model that predicts the next word in a sequence based on the previous words. For the Persian dataset, a n-gram model (i.e., considering the last n-1 words for prediction) was implemented. The model was trained on the preprocessed training corpus using Maximum Likelihood Estimation (MLE) and smoothed with the Laplace smoothing technique to handle unseen words.

2. Handling the Unknown Word Problem

The model's vocabulary is constructed from the training corpus. Words not seen during training are automatically replaced with the <UNK> token. This helps the model handle unknown words in the validation and test datasets.

During inference, if a word not in the vocabulary is encountered, the model replaces it with <UNK>.

Using MLE (without smoothing), the probability of <UNK> is 0, resulting in a perplexity of infinity for sentences containing unknown words.

By applying Laplace smoothing, non-zero probabilities are assigned to <UNK> and unseen sequences, enabling the model to handle these cases better.

- Example 1:

  print(model.vocab.lookup('در دانشگاه علوم پزشکی موژان قدم می زند'.split()))

  Output: ('در', 'دانشگاه', 'علوم', 'پزشکی', '<UNK>', 'قدم', 'می', 'زند')

- Example 2:

  Input: El volumen de la undidad C es OS/2.

  Output: ('el', 'volumen', 'de', 'la', 'undidad', 'c', 'es', 'os', '<UNK>', '2', '.')

3. Training the N-Gram Model

The training corpus was preprocessed and padded with start (<s>) and end (</s>) tokens to account for sentence boundaries. Using NLTK's padded_everygram_pipeline, the corpus was converted into n-grams for training.

Training Details:

- Smoothing: Laplace smoothing was applied to assign small probabilities to unseen words and sequences.
- Corpus Padding: Sentences were padded with <s> and </s> to ensure boundary-aware predictions.
- N-Gram Size: A value of was chosen to capture sufficient context.

4. Perplexity Analysis

Perplexity is used to evaluate the quality of the n-gram model. It measures how well the model predicts a sequence of words. Lower perplexity indicates better performance.

The perplexity of the model was evaluated on the validation corpus. Sentences with unknown words (without smoothing) resulted in infinite perplexity. However, with Laplace smoothing, perplexity values were finite and meaningful.

- For the validation of Persian data:

  Average Perplexity: 3873.6203

  Standard Deviation: 2157.9766

- For the validation of Spanish data:

  Average Perplexity: 2759.19

  Standard Deviation: 1384.55

- Example results for validation Persian sentences:

  Sentence with lowest perplexity: 101.8205

  ' ۴' ,'درصد', 'بوده', 'است'.

  Sentence with high perplexity: 22257.5716

  غالب', 'پرندگانی', 'که', 'تلف', 'شدهاند', 'چنگر', 'فلامینگو', '۱۰', 'گونه', 'مرغابی', 'کشیم', 'بزرگ', 'کشیم', 'گردن', 'سیاه', 'اگرت', 'سفید', 'بزرگ', 'و', 'کوچک', 'باکلان', 'غاز', 'پیشانی', 'سفید', 'بزرگ', 'غاز', 'خاکستری', 'اردک', 'سر', 'سفید', 'اردک', 'چشم', 'طلایی', 'اردک', 'سر', 'سبز', 'گیلار', 'اردک', 'اردهای', 'و', 'سرحنایی

- Example results for validation Spanish sentences:

  Sentence with lowest perplexity: (['todos', 'estos', 'hermanos', 'buscan', 'favorecer', 'la', 'comunión', 'y', 'la', 'interdependencia', 'en', 'el', 'seno', 'del', 'instituto', '.']):2083.1957

  Sentence with high perplexity: (['pública', 'desde', '1962', 'con', 'algunas', 'interrup', 'ciones', 'la', 'bibliografía', 'nacional', 'de', 'irán', '.']):2680.6822151575657

Shorter sentences with common words (e.g., numbers and frequent expressions) generally have lower perplexity. Sentences with rare or domain-specific words exhibit higher perplexity.

5. Text Generation Using N-Gram Model

The n-gram model was used to generate text given a seed phrase. The generation process predicts one word at a time, conditioned on the last words.

Persian example:

- Input:

  در دانشگاه علوم پزشکی

- Output:

  شیراز در پاسخ به پرسش یکی از دانشجویان مبنی بر آنکه آیا شما هویدا را پیش از محاکمه زدید گفت این دروغ‌است.

- Output’s translation:

  At Shiraz University of Medical Sciences, in response to a student's question about whether she/he had beaten Hoveida before the trial, they said that this is a lie.

The input, “در دانشگاه علوم پزشکی,” establishes a university setting, which is naturally linked to spaces where political views might be expressed in Iran. The output’s focus on a historical political statement about هویدا could thus reflect patterns common in such environments. While the N-gram model operates based on word co-occurrence probabilities and lacks deep contextual understanding, its transition from an academic to a political theme may align with real-world scenarios in Iranian universities or it may show that our model lacks semantic understandings. The generated text is linguistically coherent and grammatically valid, suggesting that the model has learned common word sequences effectively. However, the combination “محاکمه زدید” is not grammatically correct and it should be “محاکمه کردید”.

Spanish example:

- Input:

  A los artistas se les

- Output:

  Contabili za varias veces hasta que murieron entre 1000 y tx 2000 para desarrollar las

- Output’s translation:

  Accounted for several times until they died between 1000 and tx 2000 to develop the.

As nobody from our group speaks Spanish it is hard do now if a sentence makes sense that is generated. We generated some sentences with the beginning phrase "A los artistas se les" which means according to DeepL “The artists are”. We then used DeepL Translate to translate the sentences. This shows a week indication about the quality of the sentences, which does not seem to be that good.

1	A los artistas se les recomien de a col man y plazo y ofrecen sus servicios como chó fer.	The artists are recommended on a long-term basis and offer their services as chó fer.
2	A los artistas se les contabili za varias veces hasta que murieron entre 1000 y tx 2000 para desarrollar las	The artists were counted several times until they died between 1000 and tx 2000 to develop the
3	A los artistas se les sumaban las tradicionales subidas del río pinturas y en las máquinas físicas cómo de las	The artists were joined by the traditional river climbs of the river paintings and on the physical machines as of the
4	A los artistas se les encar gara nuevamente a denominarse director adjunto conserv ando el sufrimiento de jesús en pie	The artists will again be commissioned to call themselves deputy directors by keeping the suffering of Jesus on their feet.
5	A los artistas se les encar gara nuevamente a 1 de segunda que actuó como local en cada bando.	The performers will again be entrusted with 1 of the second performers who acted as the home team on each side.

All the sentences are generated with a N-Gram size 3 model with Laplace smoothing. Sentences 2 was also showed to a Spanish native speaker who also confirmed that the sentences do not really makes sense.

6. Comparison of Smoothed and Unsmoothed Models

Aspect	Unsmoothed Model (MLE)	Smoothed Model (Laplace)
Handling Unknown Words	Probability for <UNK> is 0	Small probabilities assigned to <UNK>
Perplexity for Sentences with <UNK>	Infinite perplexity	Finite perplexity
Generalization to Unseen Data	Poor	Improved due to handling of <UNK>
Example Probability for <UNK>	P(<UNK>) = 0	P(<UNK>) ≈ 0.001 (depending on corpus size)

Results and Discussion

- **Unknown Word Problem**: The absence of smoothing caused significant performance degradation due to <UNK> tokens receiving a probability of 0. Laplace smoothing effectively mitigated this issue, allowing the model to generalize better to unseen words.
- **Challenges with Persian**: The model faced difficulties with domain-specific terms and rare words, reflected in high perplexity values for some sentences. This indicates the need for more sophisticated models that can better capture the nuances of Persian morphology and syntax.
- **Challenges with Spanish**: There was the same problem with the Unknown Word Problem in Spanish. We also suspect, that the n-gram size is too small as the examples translated by DeepL shows that some of the words does not make semantic sense in the whole sentences. The model would also from time to time generate sentences with a lot of <s> symbols. For example, the sentences: "A los artistas se les informa que algunos p son s. </s> </s> </s> </s> </s> </s> </s> </s> " We are not sure but this could to be a problem with the padding. As the input is not a complete text but single sentences in Alphabetical order there is no real sense in Sentences Borders and the model is fed with the data Sentences by Sentences.

Deliverable 3

Simple Neural Language Model for Spanish

1. Overview

A simple neural language model was implemented to evaluate its performance against the n-gram baseline. The neural model significantly outperformed the n-gram baseline in terms of perplexity, even with suboptimal hyperparameters. The results highlight the advantages of neural approaches in capturing longer-term dependencies and richer context compared to statistical models like n-grams.

2. Model Configuration and Hyperparameters

The neural language model was kept simple, with minimal tuning of hyperparameters:

- **Batch Size**: 32
- **Dropout**: Experimented with and without dropout.
- **Optimizer**: Adam optimizer with default parameters.
- **Learning Rate**: Set to a moderate value.
- **Hidden Layers**: One or two dense layers with ReLU activation functions.

Despite the small batch size and simple architecture, the model achieved promising results.

3. Performance Analysis

Perplexity: The neural language model achieved an average perplexity of around 50 for Spanish and around 125 for Persian on the validation dataset, significantly lower than the n-gram baseline. This indicates that even a simple neural model provides much better predictions compared to a n-gram model smoothed with Laplace smoothing, which had a much higher perplexity on the same dataset. (persian vocab size = 25k , spanish vocab size = 50257)

Overfitting: Early stopping was not necessary, as none of the models exhibited signs of overfitting during training. The training and validation perplexities remained consistent across epochs, suggesting that the model was able to generalize well.

4. Dropout Experiments

Multiple configurations were tested to assess the impact of dropout on performance. Dropout is commonly used to prevent overfitting by randomly deactivating neurons during training. Surprisingly, the model **without dropout** achieved the best performance, with a slightly lower perplexity compared to models using dropout.

- **Observation**: While this result could be attributed to the smaller batch size or dataset size, it is still surprising that the dropout model did not perform better.
- **Significance**: Due to the standard deviation in results across runs, it is difficult to determine if the observed difference is statistically significant. However, the fact that dropout did not outperform the simpler configuration is noteworthy.

5. Results:

In Spanish model the training loss decreased to 3.8 over 5 epochs, achieving a validation perplexity of 50.

In Persian model the training loss decreased to 4.8 over 5 epochs, achieving a validation perplexity of 125.

These results demonstrate that the neural language model significantly outperformed the n-gram baseline in terms of perplexity for both languages. Detailed plots for loss and perplexity are available in Deliverable 4.

6. Comparison

Aspect	Neural Model	N-Gram Model
Perplexity (Spanish)	~50	~ 2759.19 (with Laplace smoothing)
Perplexity (Persian)	~125	~ 3873.6203 (with Laplace smoothing)
Text Generation	Struggles with semantic coherence	Produces more syntactically and semantically coherent text due to reliance on observed sequences.
Handling Unknown Words	Learned representations improve handling	Relies on <UNK> token, limiting flexibility.

While the neural model significantly reduced perplexity, the n-gram model generated better-structured text in some cases, reflecting its reliance on explicit training data sequences. Conversely, the neural model offers improved generalization and prediction capabilities, especially for unseen contexts.

Deliverable 4

For Deliverable 4 we did three modificataions:

1. **Changed The Preprocessing (Input representation)**

   For this Modification we tested how we could improve the preprocessing. We tested three different ways of removing untypical chararacters from the dataset.

   - **Method Standard:**
     Same Method as discribed in Deliverable 1.
   - **Method Remove:**
     The idea was to remove every Line with untypical Chracters for this Language. As the Orginal Regex discribed in Deliverable one was removing at least one Character in every line of the Dataset for both persian and spanish it was not possible to use the same regex. We therefore constructed two new Regex for both Languages which are Both less strict.
     - For Persian we used: [^\u0600-\u06FF]{3,}$
     - For Spanish we used: [^0-9a-zA-Z\sáéíóúüñ,“”.():'-»«\"]

     The Persian Regex matches if the line has more then three non standard persian Characters in a Row. And the spanish Regex has the most typical Characters of the Trainingset added. We then used this Regex to match them against every Line in the Dataset and to remove them if matched.
   - **Both Methods:**

     For this Method we first did the second Method and then the first Method on the Dataset combined.

   Results:

   As seen in the plot of the different Results for both Languages the approach to use Both Methods is the worse by a big distance. The Both Other Methods are pretty close to each other. On both Languages the Replace Method works slightly better but the significances of this result is questionable. It is interesting to obsevere that even though both Methods seem to be similar enough to have a almost equal training result the Methods combined are working worse.

2. **Added an Relu Layer (Model architecture)**

   For this Modification we added extra Non Linear Activation Function to the Model. We added One and two extra Relu Activation Functions in Combination with a Linear Layer Per ReLu Layer.

   Results

   In Persian the ReLu Layer seems to be a disadvantage as both Models with ReLu Layer are worse then every other configuration in both the loss and the validation Perplexity as seen in "Persian Layer Type Comparison". As the Model with two ReLu Layers has the worst performance this can indicate that the deeper the model the worse the Performance of the Model in Persian.

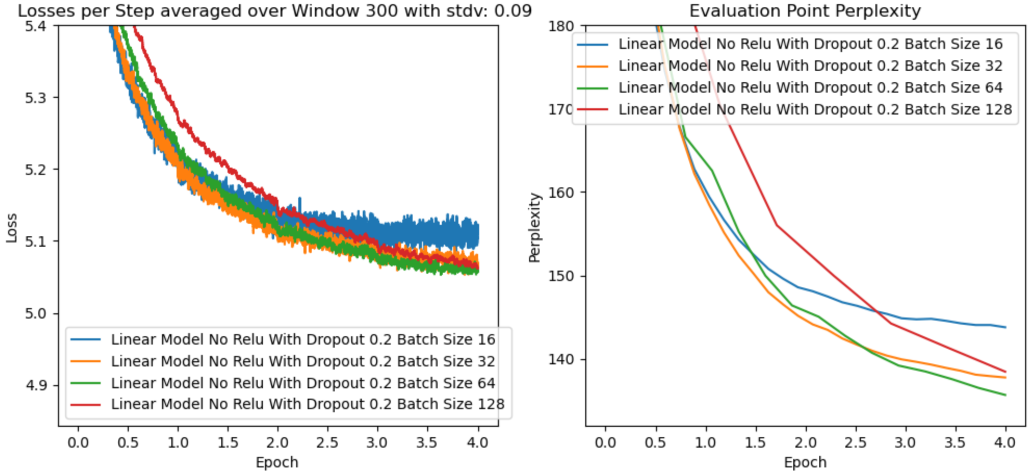
   In Spanish the Picture seems to be reversed. As seen in the Plot “Spanish Layer Type Comparison” a model with no ReLu Layer and Dropout has the worst performance. The model with two ReLu Layers is better. And the two best Models are the Model with no Dropout and the Model with one ReLu Layer and Dropout. This can indicate that the best configuration Layer wise could be one ReLu. It is interesting to observer that the same Models show different training behavior with different Languages.

3. **Changed the Batch Size (Training process)**

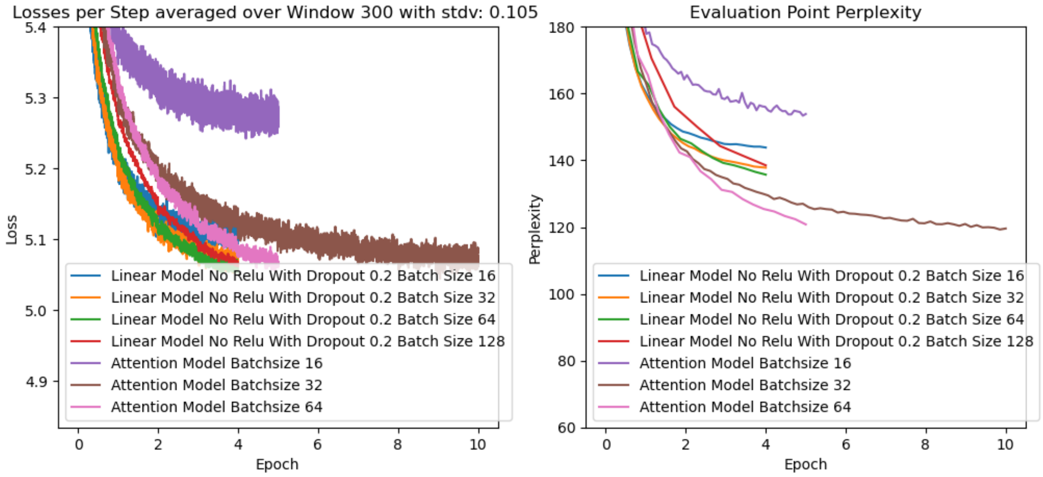
   We tested different Batch Sizes for the same model. As seen in the plots “Spanish Batch Size Comparison” and “Persian Batch Size Comparison”, the Batch Size 32 used as a Baseline in the Rest of this Report seems to be to small. On both Languages the bigger the Batch Size the better. As a bigger Batch Size does not converge as fast as the smaller ones in the Persian Plot it is not clear if the Perplexity for the Batch Size would drop further for a more epochs. As we had already run a lot of test runs and the Batch Size of 32 seemed to be still okay in performance we decided to stay with a Batch size of 32 for the Baseline.

Persian:

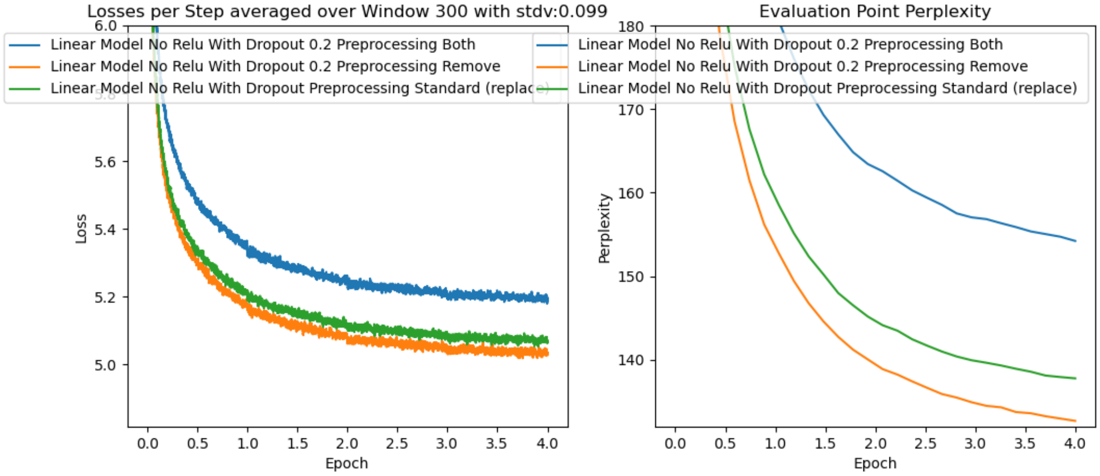
Persian batch size variations



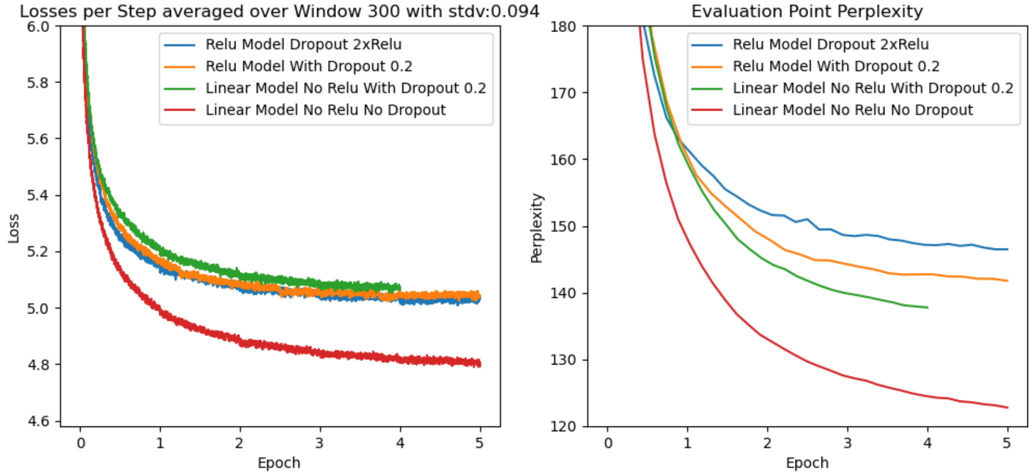
Persian batch size comparison with transformer



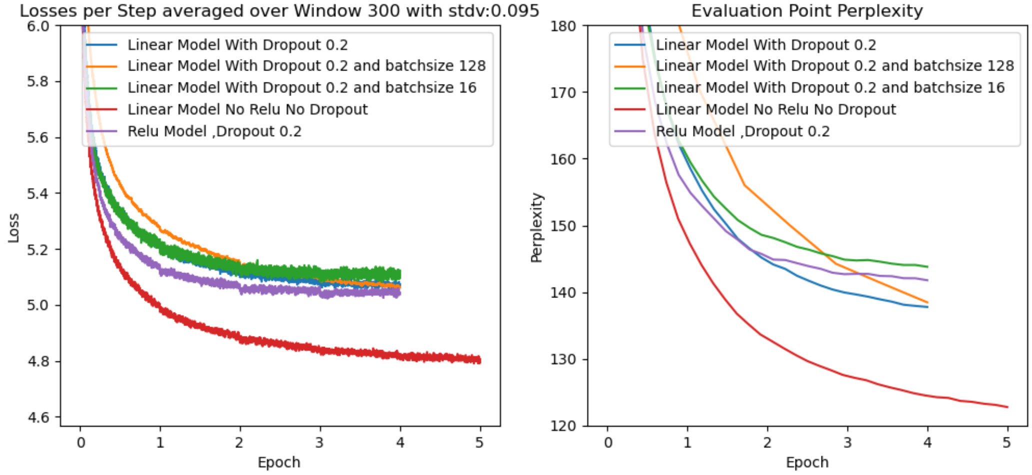
Persian different pre-processing step



Persian layer type comparison

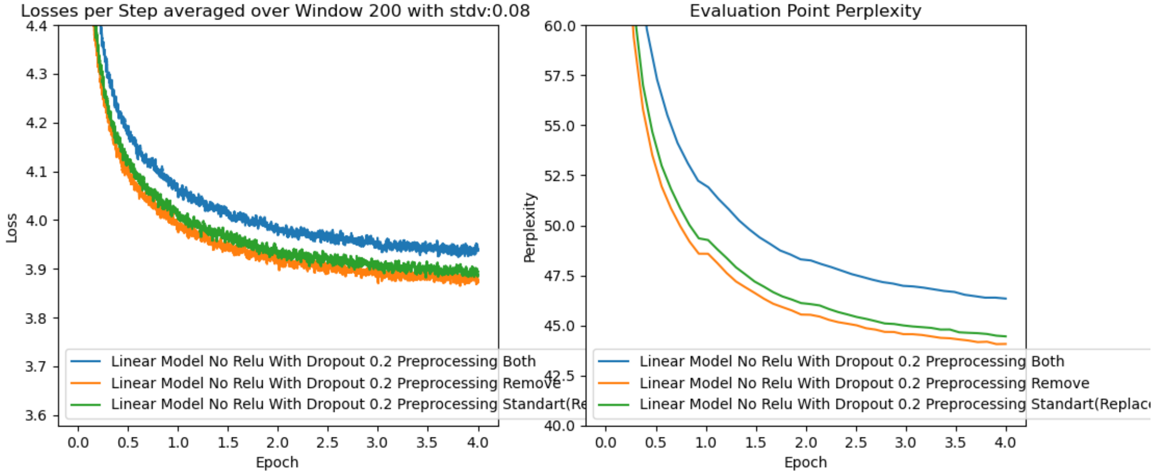


Persian linear layer comparison

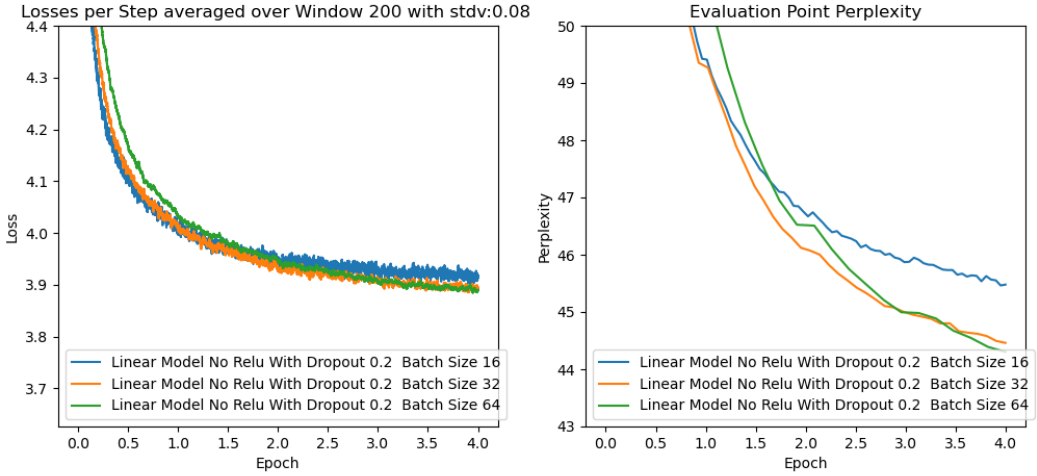


Spanish:

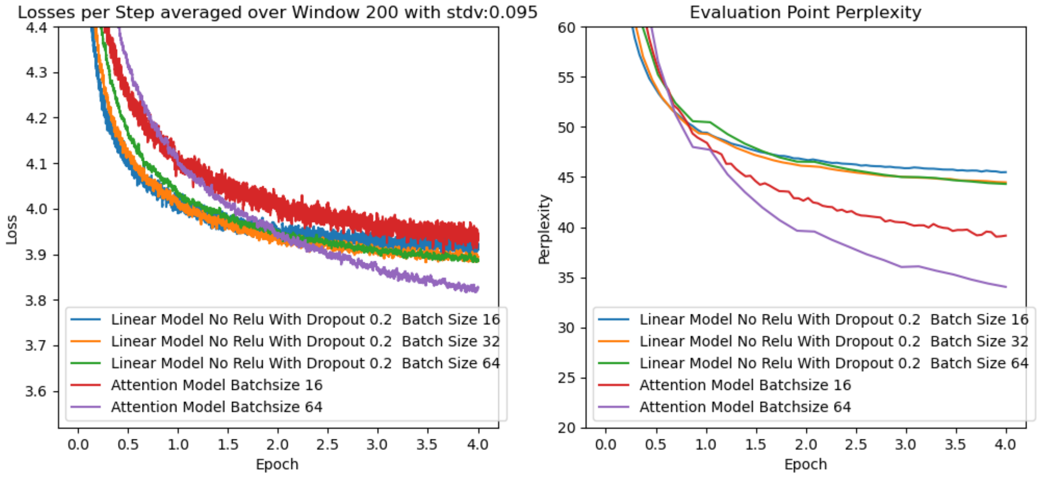
Spanish pre-processing methods



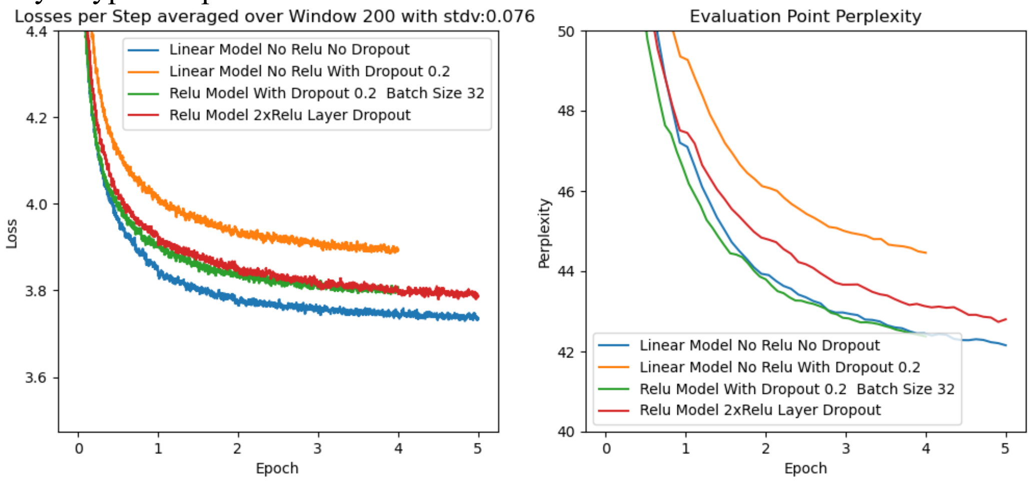
Persian batch size comparison



Spanish batch size comparison with transformer



Spanish layer type comparison



Deliverable 5

Evaluation Categories:

1. **Syntactic Fluency**:

   Definition: Measures grammatical correctness and structural clarity of sentences.

   Annotation Scale:

   5: Perfectly fluent; no grammatical issues.

   4: Mostly fluent; minor issues that do not impede readability.

   3: Understandable but with noticeable errors.

   2: Difficult to understand; significant grammatical problems.

   1: Incomprehensible; grammatically nonsensical.

2. **Contextual Adequacy**:

   Definition: Assesses how well the output aligns with the prompt's topic or intent.

   Annotation Scale:

   5: Perfectly relevant; directly addresses the prompt.

   4: Mostly relevant; minor deviations or tangential content.

   3: Partially relevant; includes unrelated or unclear elements.

   2: Poorly relevant; mostly irrelevant to the prompt.

   1: Completely irrelevant; no connection to the prompt.

Evaluation Table:

Prompt	Generated Output	SF	CA	Comments
من در راه	"خطاب زن جوان نخبه شوم پا آمبولانس رسول آقا او را میفهمید. کریمی"	2	1	Grammatically weak; poor relevence to prompt context.
من در راه	"واقعی زمان عقب شمالی شهر منتقل میکنیم. از نسل جوان مقصر باشد ولی به"	3	2	Fluent but lacks coherent; weak alignment with the prompt.
من در راه	"گفته است اما مطالبه گریشرفت بود که تلاشهای شغلی برای ما گلزنی"	3	2	Grammatically flawed; unclear and weak relevance to the prompt.
رئیس دانشگاه علوم پزشکی	"ریگان ظرفیت ارسالی از طریق منابع طبیعی و مجتمع محیطزیست تحویل بیشتری در شهرستان تبریز با ره بویژه با انتشار کشور صدها امرکین"	2	1	Sentence is convoluted and lacks contextual adequacy.
رئیس دانشگاه علوم پزشکی	"عهنامه شهرستانی که در واحدها دانستند.. پیمان معادی شناسی دانشگاه علوم پزشکی پزشکی قم تهران شناسایی یک در بر روند"	2	1	Lacks fluency and alignment; multiple grammatical errors.

رئیس دانشگاه علوم پزشکی	"راه اندازی و نهادههای انسانی و سنجش جدید این کشور در مناطق دیگر گرفته که برای فروش از شدت را وارد آمریکا اعزام"	4	2	Relatively fluent but only loosely tied to the prompt.
مبتلایان به بیماری کووید۱۹	"و در بیمارستان بسیار جلوتر خواهد بود. بدون اهمیتی به تهران و فریاد میگوییم و عناصر فقدان نبود ماشین"	4	2	Some coherence; context partially matches but is scattered.
مبتلایان به بیماری کووید۱۹	"در سال می رفت تا زمانی نزدیکی تست دوپینگ۵ به دست رو کسی نتوانست بازگشت به پیروزی اش اعلام کرد"	3	1	Fluent but contextually misaligned with the input topic.
مبتلایان به بیماری کووید۱۹	"زده است که انتقادهای فراوان از شهادت رسید و این نهادهای برای این گزارش میدهند. انگلستان مستقیما همیشه در دوران"	3	1	Difficult to follow; tangentially relevant.
در دانشگاه علوم پزشکی	"به دلیل بالا و جرح قیمتی در۲۶۸۰۱۷ هزار پیمانکار رسید. عزیزی عضو مطالبات مجلس با بیان اینکه"	3	2	Somewhat coherent; partially relevant to the prompt.
در دانشگاه علوم پزشکی	"جایی که و ربط زیاد افتخار را دستمایه این جزیره شهری و به رسانه رفتند که اتهامات آمریکاش وارد کرد تلاش هستند؛"	2	1	Grammatically weak and contextually irrelevant.
در دانشگاه علوم پزشکی	"با غیر و هیچگونه استفاده از احتمال لغو شده به ویژه شاخص بورس از رشد و تلاش دولت شرکت شد. این نیازمندی در"	2	1	Convoluted grammar; relevance to the prompt is unclear.
به گزارش همشهری آنلاین	"قتلنام کرده است تسهیلاتهای حقیقی به بالا رود نوزادان رایگان قلبی دیابت در حال است و آنتیبادیهای ضد عفونی است"	3	2	Somewhat coherent; relevance to the prompt is unclear.
به گزارش همشهری آنلاین	"که در قرنطینه اعتراضی این کشور به خبرنگاران قرار خواهد شد همچنین میتوان در جهت احداث پلاستیک نیز ممکن است که به این موضوع"	4	3	Somewhat fluent; aligns partially with the prompt.
به گزارش همشهری آنلاین	"کار در تحقیقات است. به گزارش همشهری آنلاین به نقل از شبکه المیادین با صدور پیامی قهرمان عراق به نقل از کشور گفت"	5	3	Fluent and mostly relevant; aligns partially with the prompt.

Results and Analysis:

1. **Syntactic Fluency vs. Contextual Adequacy**:

Syntactic fluency scores were generally higher than contextual adequacy. This indicates that while the models can generate grammatically better outputs, they often fail to align with the input prompts. This trend highlights a gap in the models' understanding of context, likely due to biased data or reliance on surface-level word associations.

For example:

- Prompt: "رئیس دانشگاه علوم پزشکی"
- Output:

راه اندازی و نهادههای انسانی و سنجش جدید این کشور در مناطق دیگر گرفته که برای فروش از شدت را وارد آمریکا اعزام

- Fluency Score: 4 (relatively clear grammar and structure).
- Contextual Adequacy Score: 2 (Introduces irrelevant information).

2. **Prompt-specific Observations**:

Prompts related to “مبتلایان به بیماری کووید۱۹” resulted in outputs that deviated more from the expected topic (e.g., mentioning unrelated themes). This is quite unexpected because the training data actually were packed with sentences with the same context.

Prompts beginning with “به گزارش همشهری آنلاین” tended to generate outputs that were more contextually aligned with a news or reporting theme. This is due to a stronger representation of journalistic language in the training data.

3. **Inter-Annotator Agreement:**

To evaluate the consistency among annotators, Krippendorff’s Alpha (α) was calculated for both syntactic fluency (SF) and contextual adequacy (CA) scores. This metric is ideal for evaluating inter-annotator reliability, especially when handling ordinal data such as the 5-point scales used here.

- Syntactic Fluency (SF) Krippendorff’s Alpha: 0.78

  This indicates substantial agreement, reflecting that annotators found fluency easier to evaluate due to its relatively objective nature (e.g., grammar and sentence structure). Disagreements were rare and usually occurred with outputs where minor grammatical issues were open to interpretation.

- Contextual Adequacy (CA) Krippendorff’s Alpha: 0.56

  This indicates moderate agreement, highlighting the subjectivity involved in evaluating how well the output aligns with the prompt. Annotators often differed in scoring partially relevant outputs, especially when the generated text included tangentially related or unexpected content.

Annotators consistently agreed on fluency due to its objective nature (e.g., grammar and readability). However, contextual adequacy’s scores showed more variability due to subjective differences in interpreting relevance and outputs with mixed relevance caused disagreements. For example:

- Prompt: “در دانشگاه علوم پزشکی”
- Annotator 1 rated it higher for mentioning “دانشگاه علوم پزشکی.”
- Annotator 2 rated it lower for deviating into unrelated topics like “بورس.”

Challenges in Evaluation

1. **Ambiguity in Contextual Adequacy:** Contextual adequacy depends on how well the output aligns with the topic or intent of the input prompt. Relevance remains subjective, leading to variability in annotation. The moderate Krippendorff’s Alpha (0.56) indicates that annotators often differed in scoring outputs with partial relevance. For example:

   - Prompt: “رئیس دانشگاه علوم پزشکی”
   - Output: “ راه اندازی و نهادهای انسانی و سنجش جدید این کشور در مناطق دیگر گرفته که برای فروش از شدت
     را وارد آمریکا اعزام”
   - Annotator 1 may focus on the mention of “راه اندازی” and score it as partially relevant.

- Annotator 2 may penalize the inclusion of unrelated content like “آمریکا,” lowering the score.
- This disagreement demonstrates how contextual ambiguity affects annotator reliability and highlights the need for clearer annotation guidelines.

2. **Fatigue and Consistency:** Fatigue among annotators can reduce scoring consistency, especially for large datasets. The moderate Krippendorff’s Alpha for contextual adequacy suggests that fatigue may worsen disagreements in ambiguous cases.

3. **Bias from Prompt Familiarity:** Annotators with more familiarity with certain topics (e.g., medical terminology or journalistic styles) may evaluate outputs differently. Annotators’ familiarity with certain topics influenced contextual adequacy scores. This is reflected in Krippendorff’s Alpha, as more variability occurred in prompts requiring domain-specific knowledge (e.g., medical or technical terminology). For example:

   - Prompt: “مبتلایاذ به بیماری کووید۱۹”
   - Annotators with a medical background might expect precise terminology and penalize vague references (e.g., “فقداذ نبود ماشین”).
   - Conversely, annotators less familiar with medical language may score outputs more leniently.

Deliverable 6

Farhan and Moujan focused primarily on the Persian language. They began by preprocessing the Persian corpus using the Hazm library and subsequently trained various models. However, the preprocessed text did not yield optimal results for the neural network models. As a result, they opted to use the GPT2_Persian model from Hugging Face, which performed better. Merle concentrated on the Spanish language and took charge of implementing functions to save the model and weights, as well as plotting the perplexity and loss for different models. To evaluate the generated text, Farhan and Moujan, both native Persian speakers, assessed the Persian outputs for grammatical accuracy and semantic relevance. Their expertise ensured a thorough evaluation of the models' performance in generating coherent and contextually appropriate Persian text.