NLP Sentiment Analysis Project Report

1. **Sentiment Analysis Dataset Creation (Corpus Creation)**
   The corpus creation process follows the pipeline implemented in Corpus-Creation repository:
   - Data Collection:
     - Sources are scraped or collected from r/Ljubljana subreddit using the Reddit API.
     - Texts are filtered for relevance and cleaned from noise (e.g., HTML tags, emojis, URLs).
   - Corpus Structuring:
     - Each document is saved with a unique ID.
     - Metadata (source, date, user, etc.) is also included.
     - Outputs are stored in CSV format with structured fields.
   - Quality Control:
     - Deduplication of documents.
     - Basic validation to ensure no empty or corrupted texts.
   - For *reddit_scraper.py* code documentation see the document *Documentation for Reddit Scraper Script.pdf*

2. **Sentiment Analysis Dataset Annotation (Corpus Annotation)**
   The sentiment annotation pipeline implemented in Sentiment-Annotation-Corpus repository and includes:
   - Annotation Guidelines (see the document *Sentiment Annotation Guideline.pdf*):
     - Guidelines for annotators.
     - Clear definitions of sentiment categories (Positive, Negative, Neutral, Mixed and Sarcastic).
   - Manual Annotation:
     - Annotators read each document or sentence and assign one of the sentiment labels.
   - Inter-Annotator Agreement (IAA):
     - A corpus is annotated by multiple annotators.
     - Metric Cohen's Kappa is calculated to measure reliability (see the *cohen_kappa_score.py*).
   - Sentence Segmentation:
     - The file *divide_into_sentences.py* implements a preprocessing step that converts raw textual input into sentence-level units. This step is necessary because the sentiment annotation process in this corpus operates at the sentence level rather than on full documents or paragraphs.
     - The script takes raw text files as input and applies rule-based sentence boundary detection, primarily relying on punctuation markers such as periods, question marks, exclamation points, and line breaks. The output consists of

texts segmented so that each sentence forms an independent unit, typically represented as a separate line or entry in the output files.
    - Its role is strictly transformational: it does not alter lexical content or sentiment, but reorganizes textual data into a form suitable for fine-grained linguistic and sentiment analysis.
    - Output File: *Sentiment_Corpus_Sentences.xlsx*
        - Each row corresponds to a single sentence extracted from the original source texts.

3. Exploratory Data Analysis (EDA) for annotated Corpus

- Number of words, sentences, and documents:
    - Total documents: 137 posts
    - Total sentences: 612
    - Total words: 8121
- Time range of acquired data: 12. 10. 2025 – 1. 11. 2025
- Distribution of labels (see the code *distribution_of_labels.py*):
    - The corpus displays a discrete distribution across five sentiment categories.
    - The table below reports the number of occurrences for each category and its proportion of all assigned labels.
    - Category scale: 1 = Negative, 2 = Neutral, 3 = Positive, 4 = Mixed/Other, 5 = Sarcastic.
    - The X and Y% entries correspond directly to these computed counts and percentages.

Sentiment Label	Count	Percentage
Positive	214	19.1%
Negative	360	32.2%
Neutral	448	40.1%%
Mixed	71	6.4%
Sarcastic	25	2.2%

- Additional dataset informations (see the code *dataset_overview.py*):

The subset of annotated documents enables a structural overview of the corpus. Title and body text were merged, lowercased, and stripped of non-alphabetic material before tokenization. Sentences were segmented on terminal punctuation, retaining only minimally informative units. Word counts were extracted at both sentence and document level. This yields an average sentence length indicating the typical syntactic load per unit and an average document length indicating the expected lexical span of a post. Global tokenization supports frequency analysis: the most common unigrams capture baseline lexical repetition, while the most frequent bigrams signal early collocational tendencies. Source distribution is fixed, since all entries originate from the same subreddit. Vocabulary size reflects the number of unique types in the subset. Unique token coverage is a type–token ratio expressed as a single number. It divides

the number of distinct word types by the total number of word tokens in the sample. High values imply low repetition and a broad lexical spread. Low values imply strong repetition and a narrower lexical range. These metrics constitute the dataset's basic descriptive profile and define the limits of subsequent modeling or annotation work.

- Average sentence length (words per sentence): 13.2
- Average document length (words per document): 59.3
- Most frequent content words and n-grams:
  - top_20_content_words: [('zanima', 35), ('hvala', 35), ('lahko', 31), ('ima', 30), ('res', 30), ('ljubljani', 27), ('že', 26), ('bo', 24), ('https', 22), ('več', 22), ('ko', 21), ('ker', 21), ('tako', 21), ('kar', 21), ('vse', 20), ('kje', 19), ('imajo', 18), ('gre', 18), ('nič', 17), ('naj', 17)]
  - top_10_content_bigrams: [(('https', 'www'), 16), (('www', 'ur'), 6), (('ur', 'com'), 6), (('com', 'novice'), 6), (('novice', 'crna'), 6), (('crna', 'kronika'), 6), (('html', 'https'), 6), (('novem', 'mestu'), 5), (('vozni', 'red'), 5), (('aplikacija', 'urbana'), 4)]
- Number of documents: 137 posts on Reddit (all from the same subreddit r/Ljubljana)
- Vocabulary size: 3143
- Unique token coverage: 0.555
- Differences in sentence length across sentiment labels:

  Negative : 13.9 words
  Neutral : 12.9 words
  Positive : 11.8 words
  Mixed/Other : 13.8 words
  Sarcastic : 13.8 words
- Most frequent words per sentiment category

4. Modelling the Slovenian sentiment analysis:

The modelling the Slovenian sentiment analysis process follows the pipeline implemented in Modelling-the-Slovenian-sentiment-analysis repository:

- Objective: Identify and correct mislabeled sentiment instances in a Slovene sentiment dataset and improve classifier performance. The repository contains Python code for training and evaluation of sentiment models on a Slovene opinion corpus (*kks_opinion_corpus.csv*). Core research questions focus on label noise quantification and performance gains from cleaned data.
- **Methodology:**
  a) Label Noise Detection and Correction
     - Scripts such as *noisy_labels_kks.py* imply automated detection of erroneous sentiment labels.
     - It is a corpus quality diagnostic tool that uses a simple, interpretable model to surface annotation errors, inconsistencies, and borderline cases for human review.
     - Multinomial logistic regression trained on TF-IDF features.

- Classifier: sklearn.linear_model.LogisticRegression with softmax over three classes (positive, negative, neutral), trained via maximum likelihood.
- Pipeline: TF-IDF → logistic regression.

b) Model Training
   - Zero-shot evaluation scripts assess generalization without task-specific training.
     - Method: zero-shot prompt-based sentiment classification.
     - *zero_shot_performance_gams.py*: GaMS-2B-Instruct is a decoder-only causal language model (GPT-style) with approximately 2 billion parameters. It was pretrained on large Slovene-centric corpora and instruction-tuned to follow natural language prompts.
     - *zero_shot_performance_parlasent.py:* XLM-RoBERTa (XLM-R) is encoder-only transformer and was fine-tuned for sentiment classification on parliamentary / political text (ParlaSent).
   - About GaMS-2B model: GaMS-2B (the model at cjvt/GaMS-2B on Hugging Face) is a continually pretrained variant of Google's Gemma 2 2B model with adaptations for Slovene and related languages. The publicly listed specifications include the following concrete model details from its Hugging Face page:
     - Parameters and Architecture:
       - Parameter count: 3 billion (3 B) total parameters.
       - Base architecture: Gemma 2-2B (Google's decoder-only model).
       - Model type: Gemma2ForCausalLM / gemma2 family.
     - Model Files, Precision, and Size:
       - Tensor type (full weights): F32 (32-bit floats) in the statically hosted safetensors.
       - Reported model size in HF metadata: 3 B params (file summary).
       - Quantized variants (from community GGUF builds): ~2.61 B params in some quant formats, with sizes ranging ~1.2–5.2 GB depending on precision.
     - Context & Tokenization:
       - Context/window length (for related instruct variant): 8192 tokens supported.
       - Tokenizer class: GemmaTokenizer (inherits Gemma2 tokenization).
     - Training Data and Pretraining: continually pretrained on a mix of parallel corpora and monolingual corpora covering the

listed languages, totaling ~13.6 billion tokens in second-stage pretraining.

- GaMS-2B with IFT (Instruction fine-tuning) on original KKS dataset and corrected KKS dataset: Base pretraining of GaMS-2B provides general linguistic competence; downstream usability depends on alignment via Instruction Fine-Tuning (IFT). IFT adapts a pretrained model to follow human instructions by training on ⟨instruction, input, output⟩ triples. Scripts *gams_finetune_original_KKS_dataset* and *gams_finetune_corrected_KKS_dataset* compute the accuracy and F1 score and also save misclassified instances into a CSV file.
  - KKS Corrected Dataset: 256 errors were corrected

Original → Corrected	Count
positive → negative	79
positive → neutral	23
negative → positive	12
negative → neutral	22
neutral → positive	22
neutral → negative	98
Total	256

    Out of 4,777 total instances, 256 labels (5.4%) were corrected.
- Two classifier implementations present:
  - *classifier_sloberta.py*: Fine-tuning Slovene BERT-like model (SloBERTa; transformer-based sequence classifier) for sentiment classification. SloBERTa is a transformer pretrained on large Slovene corpora. It provides contextual token representations.
  - *classifier_crosloengual-bert.py*: Cross-lingual BERT classifier leveraging multilingual representations for sentiment. CroSloEngual BERT is a BERT-base architecture pretrained jointly on Croatian, Slovene, and English. It is multilingual but regionally focused, not a generic mBERT.
  - Result files (*sloberta_results.ods, crosloengual-bert_results.ods*) capture all the instances with examples which the classifier thinks are mislabeled.
  - Files *sloberta_results_corrected_labels.ods* and *crosloengual-bert_results_corrected_labels.ods* contain human-corrected labels where the original label and predicted label did not match.
  - The script *compare_bert_results.py* compares two BERT-based sentiment analysis models (CroSloEngual BERT and SloBERTa) on the Slovenian KKS dataset, analyzing their performance and the intersection of detected label errors.

- Function *analyze_results()*: analyzes a single model: computes accuracy, macro-F1, and percentage of real errors in detected noisy instances.
- Function *get_real_errors()*: identifies instances where human_corrected_label ≠ original_label.
- Function a*nalyze_intersection()*: computes set operations (intersection, difference) on real errors between models.
- Function *create_intersection_dataframe():* merges data from both models for common errors.
- Intersection of detected noisy instances:
  - CroSloEngual BERT: 679
  - SloBERTa: 657
  - Intersection: 349

c) Evaluation
- Evaluation metrics: Accuracy and F1-score.

- **Table with results – quantitative evaluation:**

Models	Setting	F1 score	Accuracy	Comments
XLM-R-parlasent	Zero-shot	Parlasent macro-F1: 0.27178851714167 7	Parlasent accuracy: 0.68828451882845 19	Model shows a strong imbalance between metrics: relatively high accuracy (0.69) alongside very low macro-F1 (0.27). This indicates that the model largely relies on majority-class predictions and fails to capture minority classes, making its apparent performance misleading. The result suggests limited usefulness of zero-shot transfer for Parlasent without domain adaptation or supervised fine-tuning.
GaMS 2B	Zero-shot	GaMS macro-F1: 0.17080171864923668	GaMS accuracy: 0.24372384937238495	The zero-shot GaMS 2B model performs poorly on both metrics, with very low macro-F1 (0.17) and accuracy (0.24). The low accuracy indicates that the model does not even converge on majority-class predictions, suggesting weak zero-shot transfer and poor alignment with the GaMS label space. Overall, the results point to minimal task understanding in a zero-shot setting.
GaMS 2B	IFT–original data	GaMS macro-F1 (IFT original KKS dataset): 0.7543	GaMS accuracy (IFT original KKS dataset): 0.8452	Accuracy (0.85) is substantially higher than macro-F1 (0.75). This implies class imbalance in the IFT KKS dataset and uneven per-class precision/recall. The model is reliable on dominant categories but less consistent across the full label set.

GaMS 2B	IFT – corrected data	GaMS macro-F1 (IFT corrected KKS dataset): 0.7702	GaMS accuracy (IFT corrected KKS dataset): 0.8651	Macro-F1 score (0.7702) reflects improved and more balanced per-class performance compared to the original dataset. Accuracy (0.8651) shows stronger overall classification success. In comparison to original IFT results, both metrics improve, with macro-F1 rising meaningfully. This confirms that data correction improved not only majority-class fit but also class-level balance. The corrected IFT KKS dataset yields a cleaner learning signal.
SloBERTa and CroSloEngual-BERT	Classifier implementation for detection of noisy labelled instances	SloBERTa macro-F1: 0.6266 CroSloEngual-BERT macro-F1: 0.7097	SloBERTa accuracy: 0.7775 (Detected noisy instances: 657; Real errors found: 57; Percentage of real error in detected noisy instances: 8.68%) CroSloEngual-BERT accuracy: 0.8221 (Detected noisy instances: 679; Real errors found: 237; Percentage of real error in detected noisy instances: 34.90%)	CroSloEngual-BERT outperforms SloBERTa on both metrics. The macro-F1 gap is substantial, indicating better balanced performance across classes, not just better majority-class prediction. The multilingual model generalizes better for this task, likely due to broader training data and better handling of annotation variability and cross-lingual noise patterns. Both models work reasonably well, but CroSloEngual-BERT is clearly the stronger classifier for noisy label detection, especially if class balance and robustness matter.

- **Qualitative evaluation of missclassified instances (*qualitative_error_analysis.py*):**
  - Missclassified instances: 148 misclassified instances (V0 – original KKS dataset) and 129 misclassified instances (V1 – corrected KKS dataset)
  - Possible error categories for sentiment missclassification:
    - Irony/ sarcasm – example: 'Wauu, za 50 litrov bom pridobil celih 30 centov. Hvala! :)'
    - Mixed sentiment – example: 'Borili so se. ... Jaz sem z veseljem gledal... Drugi polčas malo slabša igra.'...
    - Complex syntax/ negation – example: 'Nepremičninski pok se nikoli ne bo zgodil'
    - Colloquial/ slang/ dialectal language – example: 'fak stari.... to pa je pravljica.....'
    - Domain specific – example: 'Če hočemo, da gre cena še bolj dol, bi moral sedaj tečaj €/$ narasti iz cca 1,10 kolikor je ...'
    - Short/ ambiguous – example: 'Megla bo...'
    - Rhetorical devices – example: 'Zdaj smo vsi bogati?'

- The trained GaMS-2B models and datasets are uploaded on HuggingFace:
    - GaMS-2B model finetuned on original KKS dataset: https://huggingface.co/lea-vodopivec7/gams-2b-finetuned-kks-V1
    - GaMS-2B model finetuned on corrected KKS dataset: https://huggingface.co/lea-vodopivec7/gams-2b-finetuned-kks-V0