

[urdu]rmAmiri

Assignment 3: Concepts

ExpandNet Implementation of English-to-Urdu Sense Projection

Chirooth Girigowda
girigowd@ualberta.ca
University of Alberta

Sahir Momin
smomin1@ualberta.ca
University of Alberta

1 Introduction

In this assignment we implemented sense projection on Urdu words, from English sentences using ExpandNet. We performed translation with a translator **NLLB**. We then used **dbalign** along with a Bilingual dictionary(en-ur) that we found via wikiextractor and other dictionary resources(cite all of them), with this combined dictionary and translations we get the alignment from english to urdu words on all the sentences. Finally, we project the senses of the english tokens onto the urdu lemmas based on the alignment. These 3 steps which are part of ExpandNet was used to extend the idea of ExpandNet for en-ur language pair, proving that this method can be extended to low-resource languages.

2 Data Description

To begin with, we are given a bunch of files as part of data. We have

- **se_gold_ur.tsv**: A tab-separated file containing Urdu lemmas mapped to BabelNet synset IDs. Some rows are empty (no projection possible). This serves as gold data for evaluation.
- **corebnout.txt**: A file with a filtered list BabelNet IDs, used for restricting the evaluation and for sense selection.
- **se_gloss.tsv**: A tab-separated file containing BabelNet synset IDs and their English glosses (short definitions). Used to provide sense information for evaluation and for LLM Baseline's prompt.
- **se13_sentences.tsv**: Contains the original English sentences. Each row typically corresponds to a unique sentence and its sentence ID.
- **se13_tokens.tsv**: Lists the individual tokens (word/s) in each sentence, along with their

sentence ID and token information including type, lemma, POS, raw text and instance ID. Essential for projections and alignments.

- **se13.key.tsv**: The gold-standard key for the English, specifying the correct sense(s) for each instance token in the English sentences.
- **xlwsd_se13.xml**: An XML file packaging all the above information (sentences, tokenization, and sense annotations) for convenient input to WSD systems and evaluators.

3 LLM Baseline

The LLM Baseline was implemented using the model **google/gemma-3-4b-it** where we gave it a input of a English BabelNet synset gloss (definition) and ask it to generate a single-word Urdu word for each gloss using a large language model that best fits that definition. The procedure is as follows:

The model used is **google/gemma-3-4b-it**, accessed via the Hugging Face `transformers` library. Authentication is managed by supplying an HF_TOKEN using the `huggingface_hub` API. The model is loaded locally with automatic GPU support if available, or defaults to CPU otherwise. Tokenization and text generation both leverage the `transformers` library, ensuring efficient inference. Generation is performed on-device, so no calls to external APIs are made once the model is downloaded.

The code implemented here first reads all English glosses and their associated synset IDs from the file `se_gloss.tsv`. For each gloss, it sends a prompt to the LLM requesting the best single Urdu word to match the definition. The prompt used here was

"You are a bilingual lexicon expert. Given a dictionary definition: place holder for BabelNet gloss, produce the single word in Urdu that best matches this definition. Provide

only the one Urdu word without explanations! DO NOT PROVIDE ANY OTHER OUPUT BUT THE URDU WORD!! Example (Do not include OUTPUT in your response, here INPUT and OUTPUT are only present to help you distinguish INPUT and OUTPUT, they should not be present in the your response), (Only the urdu word must be present in your response) Given INPUT prompt: You are a bilingual lexicon expert. Given a dictionary definition: "burden", produce the single word in Urdu that best matches this definition. Provide only the one Urdu word without explanations! DO NOT PROVIDE ANY OTHER OUPUT BUT THE URDU WORD!! Expected OUTPUT response from you: DO NOT REPEAT THE INPUT PRROMPT IN YOUR OUPUT ONLY GIVE THE URDU WORD!"

This prompt design was iteratively refined to minimize off-target completions and suppress verbose that LLMs might otherwise default to. The concise style and explicit example helped steer the model towards a consistent template-based output format. The model’s Urdu response is collected and stored, mapping each BabelNet ID to its Urdu word from the LLM’s response. The prompt is carefully engineered to output only a single Urdu word as the response, avoiding extraneous text, but the model outputs the input prompt along with the word hence the output is saved in a temperoroy file before post processing `urdu_projections.tsv`. The postprocessing script (`LLM_Postprocessing.py`) organizes the model’s output by cleaning the prompt from the output and storing only the urdu word as expected.

3.1 Prompt Design and Justification

- The LLM is cast explicitly in the role of a “bilingual lexicon expert.”
- The definition (English gloss) is quoted and passed in the prompt.
- The instructions demand “the single word in Urdu that best matches this definition” and ask explicitly for “only the one Urdu word without explanations.”
- Several lines reinforce the instruction not to provide any additional output, context, or explanation—only the Urdu word.
- An explicit example is given in the prompt: for the definition “burden,” the expected model

output is “” (the Urdu word for burden).

- The instructions also warn the model not to repeat the prompt or template in its output.

3.2 Number of Senses Generated per Synset

3.3 Examples, Errors, and Notable Cases

(INCLUDE EXAMPLES SOME ERRORS SEEN IN THE FILE)

4 Method

4.1 Translation

4.2 Alignment

4.3 Filtering

5 Analysis

5.1 Correct vs Incorrect Projections

5.2 Contextually Possible vs Impossible Equivalents

5.3 Sources of Error

5.4 Comparison with LLM Baseline

5.5 Automatic Evaluation

5.6 Impression

6 Conclusion

References

7 Appendix