[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 obtain 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 three steps, which are part of ExpandNet, were used to extend the idea of ExpandNet for the en-ur language pair, demonstrating that this method can be extended to low-resource languages. More information on the given data is included in the Appendex under subsection Data Description.

2 LLM Baseline

The LLM baseline was implemented using the model **google/gemma-3-4b-it**, where we gave it an input of an English BabelNet synset gloss (definition) and asked it to generate a single-word Urdu translation 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 is included in the Appendix under subsection LLM Baseline Prompt:

This prompt design was iteratively refined to minimize off-target completions and suppress verbosity 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, at times, outputs the input prompt along with the word. Hence, the output is saved in a temporary file before postprocessing. The postprocessing implemented organizes the model's output by cleaning the input prompt from the output and storing only the Urdu word as expected.

Each synset is uniquely specified by its BabelNet ID and corresponding gloss. For every BabelNet ID, our LLM baseline was prompted to generate a single Urdu word that most accurately captured the meaning of the gloss and, by extension, the synset. We deliberately designed the setup to request only one Urdu word per synset, to reduce the chance of hallucinated or verbose outputs from the model. This approach also targeted the core challenge of evaluating whether the LLM could accurately determine the primary sense conveyed by the gloss and select a fitting translation in Urdu. Although generating multiple words (such as a list of synonyms) might provide a broader sense inventory per synset, it introduces ambiguity and a higher risk of imprecise sense matching, which complicates evaluation and undermines direct sense projection. Therefore, by constraining the output to a single word, we simplified interpretation, ensured a one-to-one mapping for analysis, and established

a stricter test of the LLM's sense comprehension and lexical precision.

2.1 Examples, Errors, and Notable Cases

(INCLUDE EXAMPLES SOME ERRORS SEEN IN THE FILE)

3 Method

We followed the instructions given in the ExpandNet github page(Citation - footnote of github repo) in order to implement the ExpandNet code for projecting senses from english to urdu. This code allows only projection of BabelNetIDs and not WordNet IDs although the code was changed and experimented with WordNetIDs to ensure that it was possible to do so. The projection results from the WordNetIDs were identical to the BabelNetIDs. In the expandNet code provided, the authors perform filteration at the last step before projection where they check for a valid translation by checking if the english source token and the urdu lemma are a pair in the dictionary, however since our dictionary coverage was not extensive we missed certain english tokens from the sentences that are not present in the dictionary. To ease this constraint we make the assumption that if the english word is not present in the dictionary then we assume it is the right translation and project the senses to the urdu lemma for such pairs.

3.1 Translation

We continues to use the translation from assignment 2. We experimented with an urdu pretrained LLM **large-traversaal/Alif-1.0-8B-Instruct** (cite) and **google/gemma-3-4b-it**. Both proved to be less impressive than the translation from **NLLB** (cite). Some of the common error from the other 2 methods included wrong use of urdu words in sentences along with changed meaning of sentences. This was verified from our native speaker (Sahir), and we continued with the translations from assignment 2.

3.2 Alignment

We used multiple dictionaries that we combined to form a urdu dictionary resource to improve coverage of words. (EXplain all the methods used to extract the combined dictionary)

3.3 Filtering

The dictionary that was used to check for valid translations was the same one that we used in our alignment. We check if the pair of english source token and urdu lemma exists in the dictionary and if so only then do we project the senses onto the Urdu lemmas. However we ease this constraint by making the above assumption of correct sense projections for urdu lemmas whos corresponding english source token is not present in the dictionary.

4 Analysis

(NEED TO UPDATE THIS - RIGHT NOW LLM OUTPUT)

4.1 Sense Annotation Quality

Based on the evaluation, 100 senses annotated by ExpandNet were correct out of 732 evaluated. Many incorrect senses could still be plausible in different contexts, suggesting contextual ambiguity rather than purely incorrect mappings.

4.2 Error Sources

Most errors appear to stem from alignment and sense–translation mismatches. Recurring issues include selecting overly general senses and inconsistent mapping to Urdu lemmas. Improving sense disambiguation and enforcing stricter alignment constraints could reduce these errors.

4.3 LLM vs. ExpandNet

Compared to the manually annotated ExpandNet output, the LLM baseline performed worse. ExpandNet achieved an F1 of 10.0 (sense-level), while the LLM baseline achieved 3.1.

System	Precision	Recall	F1
LLM Baseline (Sense)	4.2	2.5	3.1
ExpandNet (Sense)	13.7	7.9	10.0
LLM Baseline (Synset)	4.2	4.1	4.2
ExpandNet (Synset)	15.9	12.6	14.1

System	Core Coverage
ExpandNet (Synset)	5.2
LLM Baseline (Synset)	6.1

5 Impression

Overall performance is low for both systems, but ExpandNet is noticeably stronger. It shows better precision and recall, especially at the synset level. Weaknesses include context confusion and sense overgeneration. Future work should focus on improved sense disambiguation, tighter alignment methods, and leveraging contextual embeddings

for Urdu. Better dictionary with higher coverage as well as better translations.

6 Conclusion

7 Appendix

7.1 Data Description

To begin with, we are given a set of files as part of the data:

- **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 of 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 the 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.

7.2 LLM Baseline Prompt

The prompt used for the LLM Baseline 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 OUTPUT BUT THE URDU WORD!! Expected OUTPUT response from you: DO NOT REPEAT THE INPUT PROMPT IN YOUR OUTPUT, ONLY GIVE THE URDU WORD!"*