Morphosyntactic Structure for Low-Resource Language Translation

Alex Kraljic, Christopher Nokes

Written: 11/18/2025, Delivered:

Agenda

- **Research Goals and Questions**
- **Methodology**
  - Morphosyntactic Analysis
  - Translation via Transformer
  - Tokenization Problem
- **Experiment Setup**
- **Results**
  - Dependency Parser Accuracy
  - Stringification Option 1
  - Stringification Option 2
- **Problems**
- **Future Work**
- **Questions and References**

Research Goals and Questions

- **How does language structure impact machine translation?**
  - Machine translators learn structure (in part) through attention.
  - Low-resource environments rarely have enough data to create efficient machine translators.
    - Not enough data to train the attention mechanism.
  - Morphosyntactic taggers require significantly fewer tokens than machine translators.
    - …but low-resource taggers are less accurate.
  - Training for tagging does not necessarily require tagged data.
  - Structure helps translation in high-resource environments.
- **Can we decrease resources required for translation training by including structural data in word embedding?**

Methodology: Overview

- **Two-step pipeline**
  - Morphosyntactic analysis
  - Translation via transformer
- **Two datasets per language pair**
  - Source language CONLLU data
  - Parallel data
- **Two models used**
  - Blank SpaCy model
  - Google T5 Small via Hugging Face
- **All done within Google Colab**

Methodology: Morphosyntactic Analysis

- **Blank SpaCy models…**
  - Take a language as input
  - Handle tokenization
  - Assigned pipelines for tasks
  - All tasks must be trained
- **What pipelines did we use?**
  - Tagger (GPOS)
  - Parser (Dependencies)
- **How did we evaluate it?**
  - LAS, UAS, and tagging accuracy

Methodology: Translation via Transformer

- **Hugging Face: Google T5 Small**
- **Simple process: load, train, and test**
- **Training allows us to use our morphosyntactic data**
    - Generate syntactic tags on parallel data
    - Use non-ideal tags for the parallel data
        - We train the translator using the real tagger
    - Why? Prepares the translator for inaccurate tagging
- **How do we get our syntax data in?**
    - Problem: tokenizer wants strings
    - SpaCy output isn’t in string form

Methodology: Tokenization Problem (1/2)

- **We need to get our syntax data in.**
- **We could create our own tokenizer**
  - Ideally, the SpaCy tagger should be part of the tokenization stage
  - But, this is a massive undertaking
  - We’d have to tweak the transformer, or make our own
    - Our tokens won’t look how a pre-trained model expects
- **We could just “stringify” our morphosyntactic data.**
  - This is much easier to implement
  - But, this puts more pressure on the transformer training
    - Needs to learn the language, and our stringification syntax

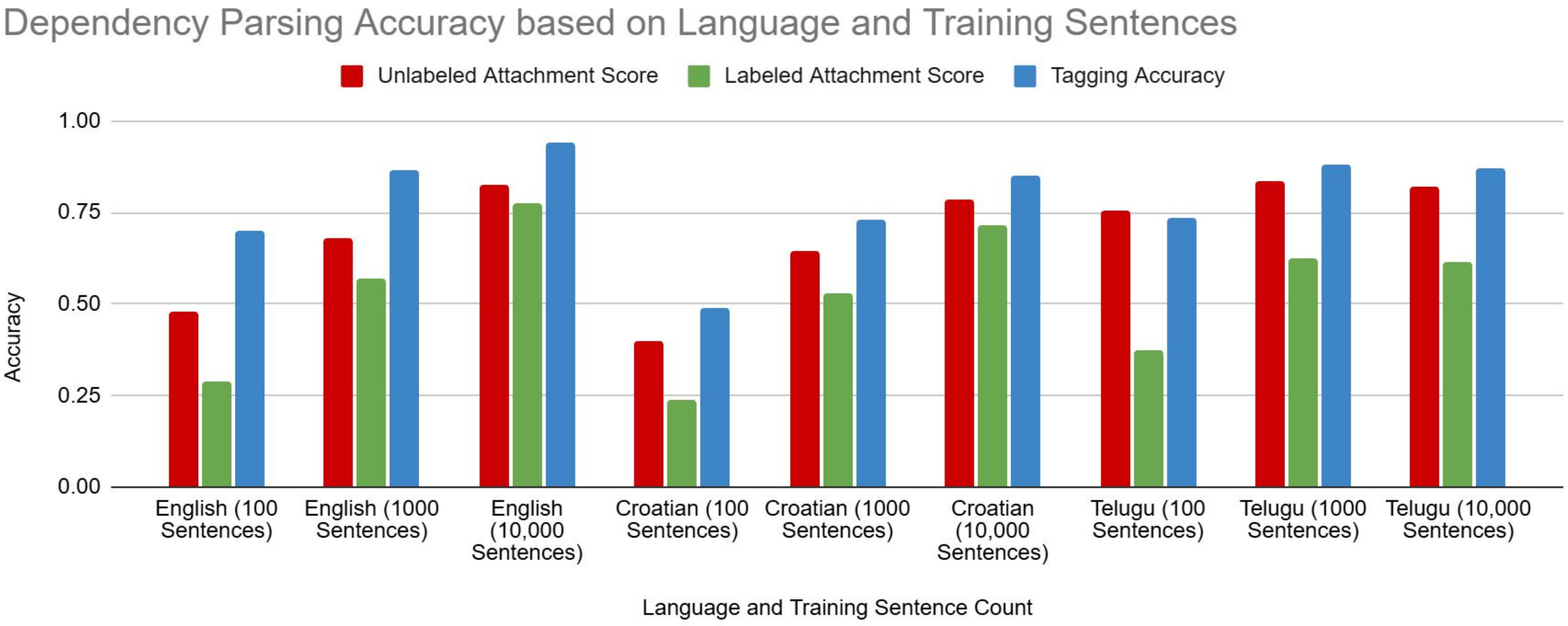
Methodology: Tokenization Problem (2/2)

- **Ultimately, we chose to take the stringification route.**
  - Time, knowledge constraints
- **New question: *how* do we stringify morphosyntactic data?**
  - Option 1: Include metadata after each word.
    - *This [PRO nsubj is] is [VER verb ROOT] a [DET …] …*
  - Option 2: Include a second metadata sentence.
    - *This is a sentence. Dependency tree: [This PRO nsubj is] …*
- **Option 1 might confuse the model if it doesn't learn the format**
- **Option 2 might just make the model discard morphosyntactic info**
- **Ultimately, we decided to test both.**

Experiment Setup

- **Two language pairs, tested bidirectionally:**
  - English ↔ Telugu, English ↔ Croatian
- **Four counts of morphosyntactic data:**
  - 0, 100, 1000, and 10,000 sentences.
  - 0 sentences = base case; no morphosyntactic tagging performed
- **Three counts of parallel data:**
  - 100, 1000, and 10,000 sentences.
- **Two different morphosyntactic stringification versions**
- **One set of hyperparameters**
- **500 sentences dedicated to evaluation via BLEU**

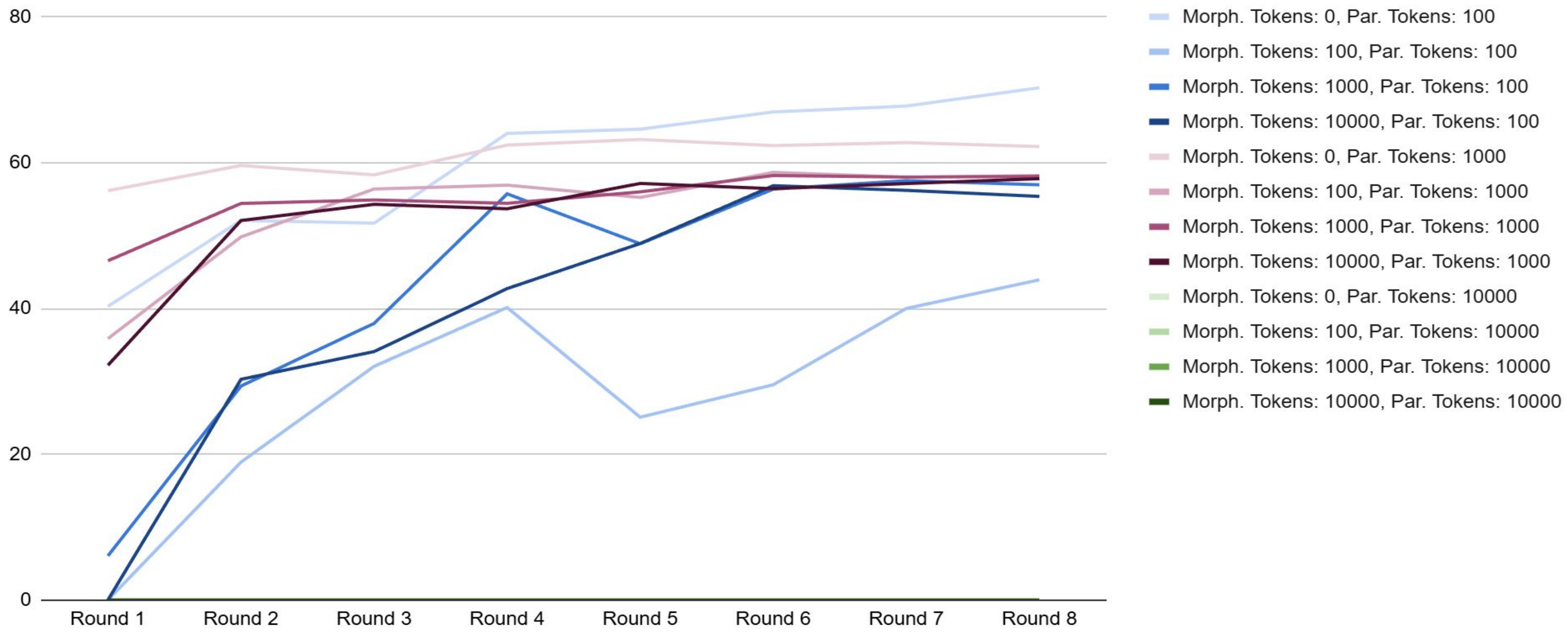
Results: Dependency Parser Accuracy



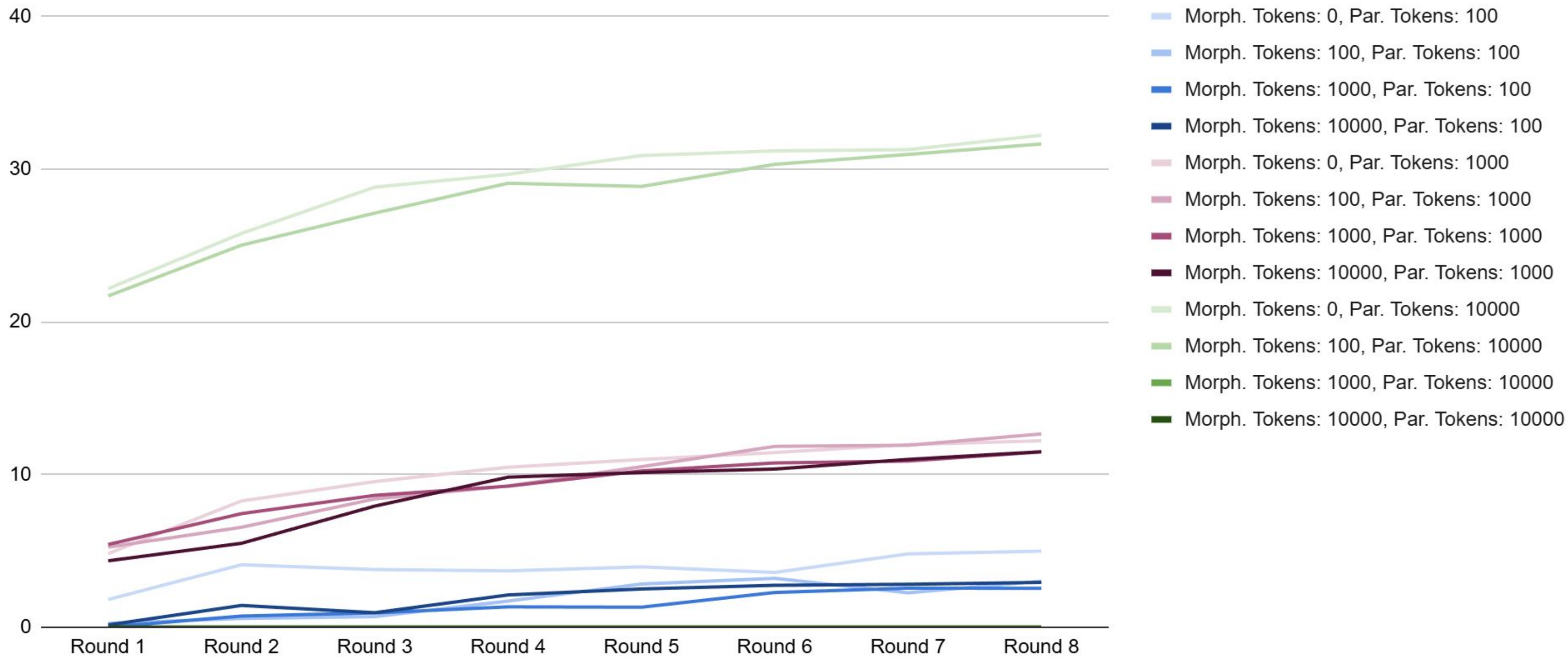
Results: Stringification Option 1

- **Reminder: Option 1 includes metadata after each word**
  - *This [PRO nsubj is] is [VER verb ROOT] a [DET …] …*
- **Results: In all cases, performed worse than control case**
  - More parallel sentences greatly improves accuracy
  - Adding morphosyntactic data in this format is detrimental
  - No correlation between tagger and translator accuracy
  - The difference isn’t massive (usually <10%)
  - On average, Croatian performs significantly worse than Telugu
- **Why might we be seeing this?**
  - As mentioned before, our formatting may be confusing the model
  - Using more data to train our formatting defeats the purpose

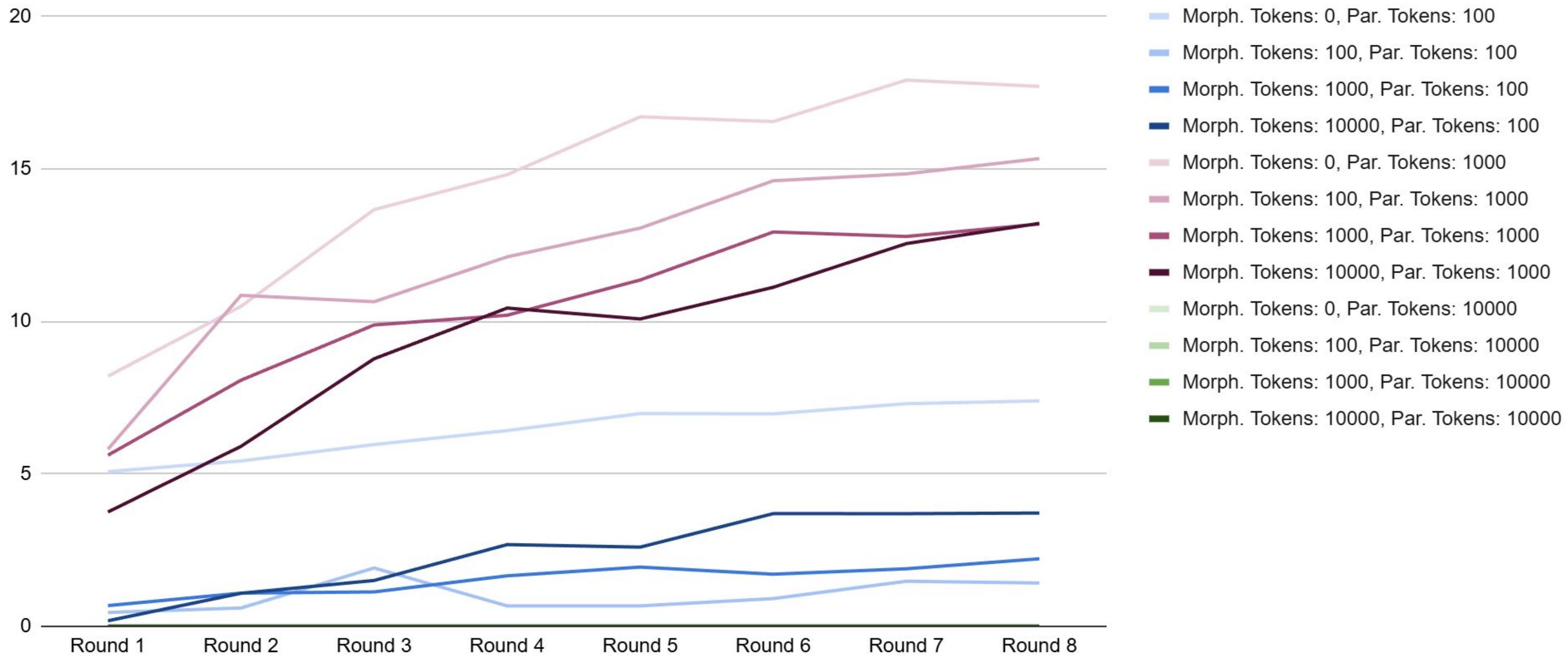
Results: Option 1, English to Telugu



Results: Option 1, English to Croatian



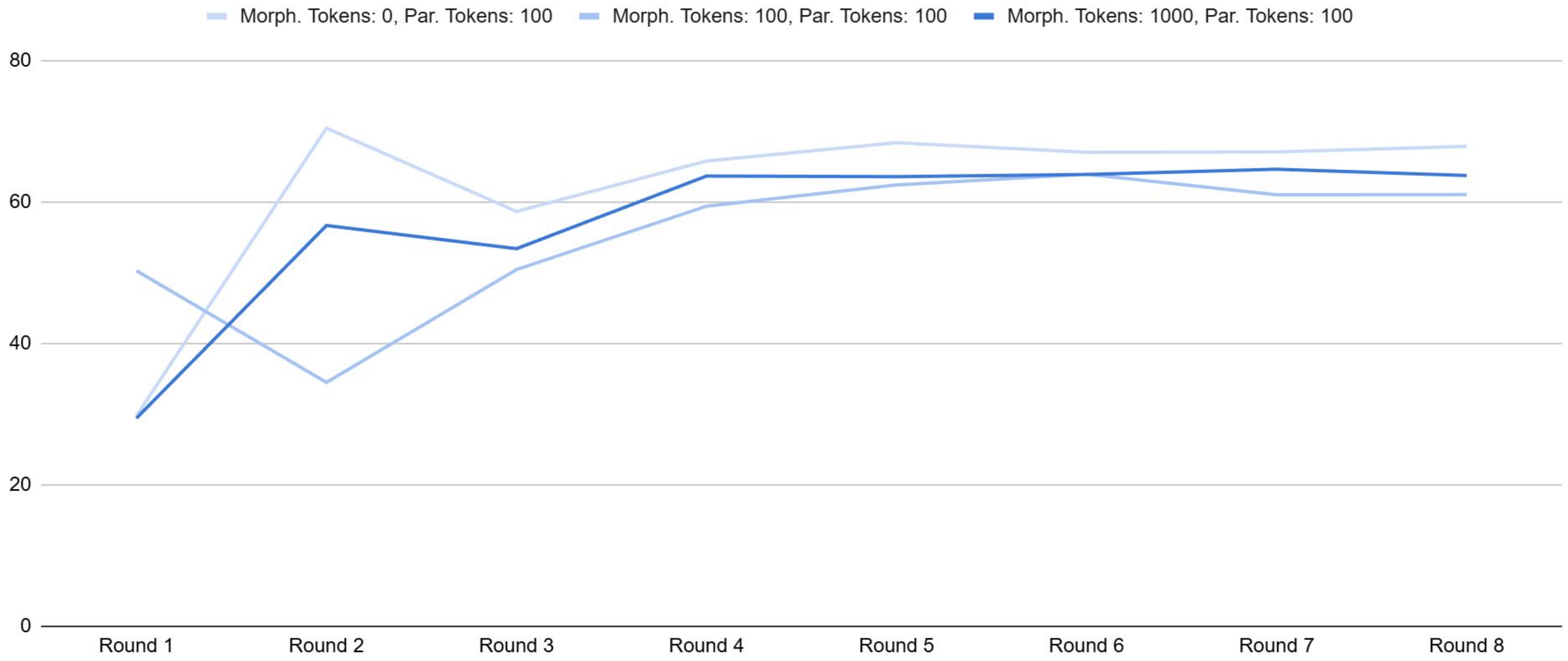
Results: Option 1, Croatian to English



Results: Stringification Option 2

- **Reminder: Option 1 includes metadata after the whole sentence**
  - *This is a sentence. Dependency tree: [This PRO nsubj is] …*
- **Results: In progress, *but* doing better**
  - Low-accuracy tags decreases accuracy by 5% versus control
  - *But* higher accuracy taggers perform better!
    - A very good tagger *may* outperform the control translator
    - This is what we are now testing for
- **Why is this performing better?**
  - The data format is simpler to understand

Results: Option 2, English to Telugu



Problems

- **The tokenization problem was a serious roadblock**
  - Lack of knowledge
  - Lack of time
- **Limited resources - Google Colab environment**
  - One test took two hours and all of the daily allotted GPU time
  - Ultimately, Colab was great for rapid prototyping
  - …but Colab didn’t work well for long-form testing
- **Limited testing due to limited resources and time**
  - Only tested one pre-trained transformer
  - Only tested one pair of datasets for each language pair

Future Work and Conclusions

- **Firstly, plans for this class project:**
  - Finish data collection for larger token counts on test set one
  - Continue collecting data for the second test set
  - Possibly run tests with more data on both test sets
  - Possibly test different languages
- **Secondly, possible continuations of the research question:**
  - Create a transformer that includes tagging during tokenization
- **Ultimately, we conclude that:**
  - Stringification *may* aid translation, depending on tag accuracy
  - Inaccurate tagging or poor stringification is actively detrimental

References

- **He, Zhiwei, et al. "Exploring Human-like Translation Strategy with Large Language Models."** Transactions of the Association for Computational Linguistics, vol. 12, 1 Jan. 2024, pp. 229–246, https://doi.org/10.1162/tacl_a_00642. Accessed 30 May 2024.
- **Hedderich, Michael, et al. A Survey on Recent Approaches for Natural Language Processing in Low-Resource Scenarios.** 9 Apr. 2021.
- **Kann, Katharina, et al. "Weakly Supervised POS Taggers Perform Poorly on Truly Low-Resource Languages."** Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 05, 3 Apr. 2020, pp. 8066–8073, ojs.aaai.org/index.php/AAAI/article/view/6317, https://doi.org/10.1609/aaai.v34i05.6317. Accessed 4 Nov. 2025.
- **King, Benjamin. Practical Natural Language Processing for Low-Resource Languages.** 2015.