Team Cantharellus at SemEval-2025 Task 3: Hallucination Span Detection with Fine-Tuning on Weakly Supervised Synthetic Data

Xinyuan Mo and **Nikolay Vorontsov** and **Tiankai Zang** *
University of Helsinki
{xinyuan.mo, nikolay.vorontsov, tiankai.zang}@helsinki.fi

Abstract

This paper describes our submission to SemEval-2025 Task-3: Mu-SHROOM, the Multilingual Shared-task on Hallucinations and Related Observable Overgeneration Mistakes, which mainly aims at detecting spans of LLM-generated text corresponding to hallucinations in multilingual and multi-model context. We explored an approach of fine-tuning pretrained language models available on Hugging Face. The results show that predictions made by a pretrained model fine-tuned on synthetic data achieve a relatively high degree of alignment with human-generated labels. We participated in 13 out of 14 available languages and reached an average ranking of 10th out of 41 participating teams, with our highest ranking reaching the top 5 place.

1 Introduction

Recent years have witnessed the rapid development of large language models (LLMs) and their applications in various fields of natural language processing (Wei et al., 2022; Zhao et al., 2023). However, content generated by LLMs occasionally contains inaccurate or fictitious information (Perković et al., 2024). The phenomenon in which natural language generation models often generate text that is nonsensical, or unfaithful to the provided source input is commonly referred to as "hallucinations" (Ji et al., 2023). It is therefore a vital task to detect and identify hallucinated content so as to improve the reliability and trustworthiness of LLM-generated content.

SemEval 2025 Task 3 (Mu-SHROOM, the Multilingual Shared-task on Hallucinations and Related Observable Overgeneration Mistakes) (Vázquez et al., 2025) proposes the task of detecting hallucination in content generated with LLMs. Different from the previous iteration, SemEval-2024 Task 6 (Mickus et al., 2024), in which the participants were asked to make binary decisions of whether a given context contains hallucination, the current task requires the participants to predict where the hallucinations occur. Specifically, the current task requires the participants to predict the spans of the hallucinated content within LLM outputs in 14 different languages.

As is shown in the results of the previous iteration of this task (Mickus et al., 2024), it is effective to fine-tune pretrained language model for hallucination detection. We therefore further extended this approach from performing binary classification tasks to predict the spans of hallucinations. We explored fine-tuning a series of Transformer-based pretrained language models (Vaswani et al., 2017), including text-to-text Transformer models (Raffel et al., 2020) and BERT-based models (Devlin, 2018), with training data created by ourselves.

Building on this method, we developed the system based and participated in 13 out of 14 languages. In addition, we also applied the approach of named entity recognition (NER) as a baseline in order to provide a more comprehensive evaluation of our system's performance. We have released our code and other relevant material on GitHub [1].

2 Background

Hallucination detection has been an extensively researched topic in recent years. One promising solution for hallucination detection is to utilize the self-evaluation ability of LLMs to judge the factual correctness of a given statement, leveraging the fact that LLMs have possessed a rich knowledge base (Li et al., 2024; Zhang et al., 2024). Many existing studies focus on the binary classification of hallucination. For instance, SelfCheckGPT (Manakul et al., 2023) is built on the idea that when an LLM is familiar with a particular concept, the responses it generates are likely to be consistent and con-

* Equal contribution, authors are listed alphabetically.

[1] https://github.com/nicksnlp/Cantharellus.git

Proceedings of the The 19th International Workshop on Semantic Evaluation (SemEval-2025), pages 1724–1736

tain similar facts. In the HaluEval 2.0 benchmark (Li et al., 2024), the hallucination detection approach is built by first extracting factual statements from LLM responses, then determining the trustfulness of these statements with respect to world knowledge by taking advantage of the vast knowledge base of LLMs. Similarly, MIND (Su et al., 2024) introduces a similar approach that leverages the internal states of LLMs for real-time hallucination detection without requiring manual annotations. GraphEval (Sansford et al., 2024) proposes a method of detecting inconsistencies with respect to provided knowledge using a knowledge-graph approach.

Prompt engineering is a crucial technique for extending the capabilities of LLMs (Sahoo et al., 2024). A commonly used strategy is few-shot prompting, which refers to the technique of constructing prompts using a small set of demonstrative input-output examples (Lazaridou et al., 2022; Ma et al., 2023). Another method that has been proved effective is chain-of-thought (CoT) prompting, which is achieved by guiding the LLM to think step by step to perform complex reasoning tasks (Wei et al., 2022; Zhang et al., 2022; Chen et al., 2023).

Named entity recognition (NER) (Yadav and Bethard, 2019) is the task of identifying named entities such as person, location and organization in text, which are often central to factual inconsistencies in LLM-generated text. Deep learning approaches have increasingly been adopted for NER and have exhibited impressive abilities across various domains and languages (Li et al., 2020; Song et al., 2021; Liu et al., 2022). Recent research also explores the possibility of integrating prompting techniques into NER tasks by utilizing LLMs (Shen et al., 2023; Hu et al., 2024). While NER is limited to identifying predefined entity types and cannot assess the broader context or relationships between entities, it can still serve as a tool for detecting potential sources of hallucinations, making it a possible referential benchmark.

3 System Overview

3.1 Fine-tuning Procedure

The goal of this task is to detect hallucinations and identify their spans, defined by character indices, within an answer generated in response to a question. One of the ways to address this is to reformulate the problem as a token classification task, where hallucinated tokens are labeled as 1 and non-hallucinated tokens as 0.

We conducted experiments on Transformer-based models to evaluate their performance on this task. Given the limited size of the validation sets (50 labeled data points per language), all base models were trained on our self-generated data (approximately 2K data points per language).

3.1.1 Data Construction

It is crucial to have sufficient data to fine-tune pretrained language models in order to enhance their performance on specific tasks. However, the training sets provided by the organizers are unlabeled and are thus unable to be used directly for fine-tuning. Therefore, we proposed a semi-automatic approach to construct labeled data by prompting state-of-the-art generative language models, specifically GPT-4o.

In order to obtain the data in a manner similar to that of the labeled validation sets provided by the organizers, we explored a combination of few-shot prompting and chain-of-thought (CoT) prompting techniques. Specifically, we utilized several data points in the labeled validation set as learning examples and guided the LLM to infer hallucinated content based on the spans marked by human annotators in those samples.

A closer examination of the generation revealed that the LLM often misidentified the span boundaries of the hallucinated words it generated. Therefore, we optimized the pipeline of data construction by prompting the LLM to identify the hallucinated words first and subsequently convert the tokens to spans. To that end a simple algorithm was applied, which would automatically iterate through the model output text, locate each hallucinated word, and mark its start and end character indices. This additional step significantly increases the quality of generated annotations.

A number of 2,371 data points in English was constructed initially for testing purposes. In addition to those, ultimately, we constructed 2000 data points for each of the 12 other languages in which we participated. The detailed prompt is shown in Appendix C.

3.1.2 Base Models and Token Classification Setup

For fine-tuning, we experimented with a diverse set of pretrained Transformer-based models (Vaswani et al., 2017), starting with monolingual architec-

tures and later transitioning to multilingual models, which support all 13 target languages, to achieve broader language coverage. We focused on the English monolingual models to compare their performance with that of the multilingual models. Table 1 and Table 2 in the Appendix B show the details of the base models.

`AutoModelForTokenClassification` classes from the Hugging Face `transformers` library (Wolf et al., 2020) are used to load each of the base models and their tokenizers. This step attaches a randomly initialized linear classification layer on top of the Transformer encoder, ensuring the model outputs token-wise predictions for binary classes (1 or 0 for hallucination or non-hallucination, respectively). The label mappings are explicitly defined through the model's configuration using `id2label` and `label2id`. The model performs sequence labeling, where each token in the input sequence is assigned a binary label based on both its own identity and its contextual information from the entire sequence.

3.2 Performance Evaluation

A system's capability to capture hallucination spans is assessed along two main dimensions: (i) the overlap between the system's predicted hallucination spans and human annotations, and (ii) the alignment in reasoning between the system and human annotators, reflected in the correlation between the confidence of system predictions and the agreement of human annotators on hallucination spans.

These two evaluation dimensions were measured using Intersection over Union (IoU) and Correlation (Cor) scores, respectively.

The IoU score quantifies the overlap between predicted hard labels and reference hallucination spans by dividing the size of their intersection by the size of their union. If neither the prediction nor the reference contains hallucinations, the score is set to 1.0.

The Cor score quantifies the agreement between predicted soft labels and reference confidence levels, which are computed as the fraction of annotators who labeled a span as hallucinated. This agreement is measured using Spearman's rank correlation (Spearman, 1987). The score ranges from -1 to 1, where 1 indicates perfect agreement, 0 signifies no correlation, and -1 represents complete disagreement.

4 Experimental Setup

4.1 Model Fine-Tuning

The fine-tuning procedure began with preprocessing the training data, aligning tokens with binary labels to indicate hallucination. Fine-tuning Stage 1 was conducted using our auto-generated training data. This step is followed by fine-tuning Stage 2 using the labeled validation sets provided by the task's organizers, either with all available sets (for multilingual models) or the validation set corresponding to the specific test language (for both monolingual and multilingual models). Model performance was assessed for both fine-tuning stages. The experimental architecture is illustrated in Figure 1.

4.1.1 Data Preprocessing

Label Alignment In all labeled datasets used for fine-tuning, hallucination spans are provided in the format `[start_index, end_index]`. We leveraged this span information to automatically generate labels for each token before feeding the training data into the base models. After tokenization, labels are assigned based on each token's `offset_mapping`: tokens whose start and end indices fall within any hallucination span receive a label of 1, while all others are labeled 0.

Data Split We used a 9:1 training-validation split on our self-generated labeled data, resulting in approximately 1,800 training samples and 200 validation samples for each of the 13 languages. These included nine announced target languages: Arabic, German, English, Spanish, Finnish, French, Hindi, Italian, and Chinese, as well as four surprise languages: Czech, Catalan, Basque and Farsi, which were revealed only after the test set was released by the organizers.

We evaluated our models' performance using the labeled validation sets provided by the organizers after fine-tuning. We chose these sets as test data due to (i) their high-quality hallucination spans annotated by human annotators and (ii) their likely similarity to the data used by the organizers for final evaluation. In contrast, our semi-automatically generated labeled data were not reviewed by native speakers and may be of lower quality. However, since labeled validation sets were not available for the four surprise languages, we generated 50 labeled data points for each of these languages for testing purpose using the same method as for our training data.

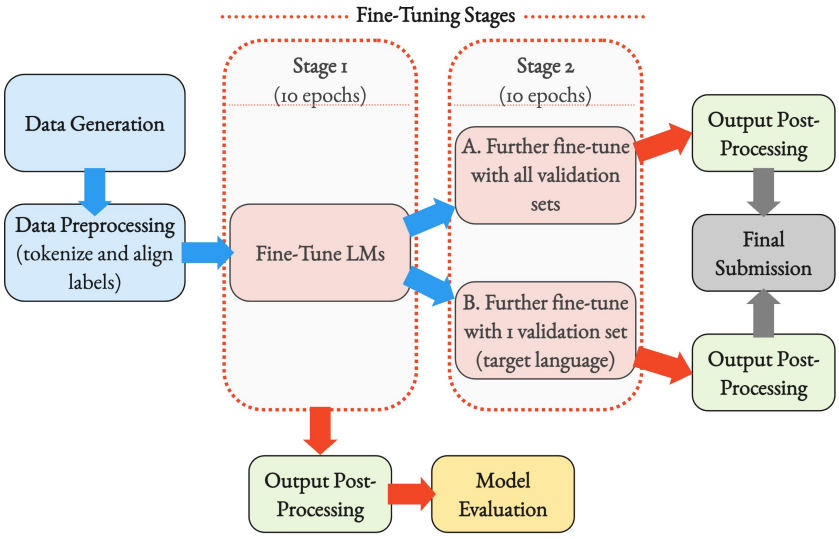


Figure 1: Fine-Tuning Pipeline.

4.1.2 Hyperparameters for Tokenization and Training

For the fine-tuning procedure, the same set of hyperparameters for both tokenization and training was applied to all base models. The same hyperparameters were applied to both fine-tuning Stage 1 and Stage 2.

Tokenizer Parameters Only the generated answers (`model_output_text`) were tokenized as input for model fine-tuning, while the inquiries (`model_input`) were not used. Tokenization included `padding` with a maximum length of `128` tokens and the application of `truncation`. Additionally, `return_offsets_mapping` was set to `True`, as the offset mapping is crucial for converting hallucination spans into token-level labels after tokenization. Details on this process are discussed in Section 4.1.1.

Training Parameters The training process was configured with a predefined set of parameters. The learning rate was set to 2e-5. Batch sizes were defined as 8 for both training and evaluation (`per_device_train_batch_size = 8` and `per_device_eval_batch_size = 8`). The number of training epochs was set to 10, as our preliminary experiments indicated that model performance plateaued around this point. Additionally, a `weight_decay` of 0.01 was applied to the training arguments.

4.1.3 Output Post-Processing

During the inference stage, the input text was tokenized and processed by the models to obtain logits for each token to be assigned to one of the two possible labels. The logits were later converted to probabilities using the softmax function, which normalized the scores along the label dimension:

$$P(y_i \mid x) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

In the formula above, $P(y_i \mid x)$ represents the probability of the i-th label for a given token, and z_i the corresponding logit.

To identify hallucination spans, contiguous tokens labeled as "1" (hallucinated) were grouped based on their start and end indices derived from the token offset mappings. Adjacent "1" labels with consecutive indices were treated as a single span. For each span, the average probability of the "1" labels was computed, providing a confidence score that reflects the model's uncertainty regarding the span's validity. This average probability, along with the start and end indices of each hallucination span, together form the soft labels. Hard labels are then derived by selecting spans from the soft labels with a probability of 0.5 or higher.

4.2 Benchmarks for Evaluation

To assess whether fine-tuning improves model performance in hallucination detection, we evaluate the predictions of the models from fine-tuning Stage 1 on the organizers' validation sets. This evaluation uses Cor and IoU scores and compares the results against three benchmarks: (i) the benchmark provided by the organizers and (ii) the performance of a pretrained multilingual Named Entity Recognition (NER) model.

Organizers' Benchmark The benchmark provided by the organizers was derived by fine-tuning the multilingual model `FacebookAI/xlm-roberta-base` (Conneau et al., 2019a) using the labeled validation sets they released. These validation sets contained 50 data points per language across 10 languages, excluding the four surprise languages.

Cor and IoU scores were computed for each test language. The base model was trained on validation sets from all languages except the test language, ensuring no test data leakage. This process was repeated for each language, yielding 10 fine-tuned models, each tailored to a specific test language. Fine-tuning was performed using the `model_output_text` and two label types, which classified tokens as either hallucinated or non-hallucinated.

NER Benchmark A close examination of the sample set shows that a considerable number of hallucinations involve proper nouns, such as names of people, places or organizations. We therefore proposed using a model fine-tuned for NER to make predictions without further fine-tuning, in order to serve as a comparison to the models fine-tuned specifically for this task.

This benchmark was created using the pre-trained NER model `51la5/roberta-large-NER` (Conneau et al., 2019b), a large multilingual language model supporting all 13 of our target languages. Trained on 2.5TB of filtered CommonCrawl data, this model is an XLM-RoBERTa-large variant fine-tuned on the CoNLL-2003 dataset for English NER. We used the model as-is, without any additional fine-tuning. As a result, it treated all named entities as hallucinations.

5 Results

Our submission included 21 models in total, with different combinations of models and training data used. These were the following:

- Multilingual models trained on the synthetic data only (26.3K data points, 10 epochs)
- Multilingual models trained on the synthetic data (26.3K data points, 10 epochs) and fine-tuned further with a single validation set for the target language (50 data points, 10 epochs)
- Multilingual models trained on the synthetic data (26.3K data points, 10 epochs) and fine-tuned further with all the validation sets (650 data points, 10 epochs)
- English language models trained on the synthetic data only (2.3K data points, 10 epochs)
- English language models trained on the synthetic data only (2.3K data points, 10 epochs) and fine-tuned further with a single validation set for the English language (50 data points, 10 epochs)

From all the combinations the models trained on the maximum amount of data showed superior results, with some minor exceptions (see Appendix D). Our best average performing model was based on `xlm-roberta-large`.

As opposed to the initial submission, when the models were trained on the generated answers only (`model_output_text`), we have conducted an additional test, and trained `xlm-roberta-large` base model again with the same parameters, but using a concatenation of the questions and the answers as the training input, separating them with an additional special token `'<@@>'`. We have used similar joint input for inference and adjusted the spans accordingly. The results of such training have outperformed all of our other approaches, with a significant increase of scores for all of the languages except English and Chinese. A comparative summary of the models' IoU scores is shown in Appendix A.

6 Conclusion and Limitations

As our participation in SemEval-2025 Task-3, we proposed the approach of fine-tuning language models with synthetic data for hallucination span prediction. The results demonstrate that our system achieved competitive scores across various languages. Our approach is proved effective as the scores rank as high as 5th in certain languages.

The fact that the model is exclusively fine-tuned on data constructed by LLM suggests that this approach is a feasible and effective strategy for the task of hallucination span prediction, particularly under the circumstance where human-labeled training data is absent. However, it is worth pointing out that LLM-based synthetic data are potentially more heavily subject to limitations compared with hand-crafted data. A closer examination suggests that the quality of the synthetic data does not match that of the validation and test sets provided by the organizers. Across various languages, the synthetic

data can often be shorter or significantly longer in output length, cover a narrower range of topics, and propose less sophisticated question-and-answer pairs than those in the validation and test sets. These problems can supposedly be addressed through few-shot prompting and more elaborate prompt-engineering. It is also important to mention, that although various efforts have been made in both prompting and post-processing to increase the likelihood that the spans correctly indicate the hallucinated texts, the final output is still prone to errors and may not match the accuracy of human-annotated data, although it was reported that models trained on data generated in a similar manner outperform models trained on real-world data in certain tasks (Li et al., 2023). In addition, it should also be noted that predominantly only one model, GPT-4o, is used for data synthesis. This lack of variation in model choice may limit the diversity of the synthetic data, which could in turn potentially reinforce the intrinsic biases of the model in question.

Future directions could include creating and utilizing training data in large quantities, as well as optimizing prompting techniques to obtain higher-quality training data from LLMs. Another viable approach would be to adopt more advanced state-of-the-art models for either data creation or fine-tuning.

7 Acknowledgments

We would like to express our gratitude to our supervisor, Jörg Tiedemann, for valuable suggestions and guidance throughout this work.

8 Individual Contributions

Xinyuan Mo: Model fine-tuning, including I. constructing python script for: 1) data preprocessing (tokenizing and aligning labels), 2) fine-tuning multiple base models, and 3) output generation and scoring (integrating code "`scorer.py`" from the organizers to generate scores), and II. executing the fine-tuning and scoring procedure on Puhti supercomputer.

Writing of the sections of the paper: System Overview (excluding the "Data Construction" subsection) and Experimental Setup.

Nikolay Vorontsov: Experimenting with various options for data construction, including: 1. Direct prompting with Gemini and GPT through APIs and output post-processing; 2. Generating question-answers pairs from given input texts (with AutoModelForQuestionAnswering, QuestionAnsweringPipeline) and refining them with a generative model. 3. Converting the labeled datasets into translated versions with the preserved labeling (with Google Translator API) and output post-processing.

Development of the system for labeling the datasets with LLMs through API (used as another baseline during development), automated post-processing of the output, union and intersections of the models' predictions.

Development of alternative fine-tuning strategies with different training parameters, including LoRA parameter-efficient fine-tuning of Llama 7B model with 4-bit quantization. Alternating tokenization pre-processing and post-processing to include both questions and answers during the training and inference stages. Setting up the environment on Puhti supercomputer, running training and prediction jobs.

Limited project management, including setting up shared repositories, communication channels, submitting the final predictions to the shared task.

Writing of the Results section of the paper, analysis of the results, creating figures, and formatting.

Tiankai Zang: Data construction, including 1) designing and testing various prompting techniques for optimal outputs, 2) overseeing the data generation process and making adjustments as needed, and 3) converting the data to desired format for fine-tuning.

Writing of the sections of the paper: Abstract, Introduction, Background, System Overview ("Data Construction" subsection), Conclusion and Limitations.

References

Jiuhai Chen, Lichang Chen, Heng Huang, and Tianyi Zhou. 2023. When do you need chain-of-thought prompting for chatgpt? *arXiv preprint arXiv:2304.03262*.

Hyung Won Chung, Noah Constant, Xavier Garcia, Adam Roberts, Yi Tay, Sharan Narang, and Orhan Firat. 2023. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining. *arXiv preprint arXiv:2304.09151*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco

Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019a. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

deepset. 2020. Roberta-base fine-tuned on squad2. https://huggingface.co/deepset/roberta-base-squad2.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing.

Yan Hu, Qingyu Chen, Jingcheng Du, Xueqing Peng, Vipina Kuttichi Keloth, Xu Zuo, Yujia Zhou, Zehan Li, Xiaoqian Jiang, Zhiyong Lu, et al. 2024. Improving large language models for clinical named entity recognition via prompt engineering. *Journal of the American Medical Informatics Association*, page ocad259.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115*.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE transactions on knowledge and data engineering*, 34(1):50–70.

Junyi Li, Jie Chen, Ruiyang Ren, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. The dawn after the dark: An empirical study on factuality hallucination in large language models. *arXiv preprint arXiv:2401.03205*.

Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. 2023. Synthetic data generation with large language models for text classification: Potential and limitations. pages 10443–10461, Singapore.

Pan Liu, Yanming Guo, Fenglei Wang, and Guohui Li. 2022. Chinese named entity recognition: The state of the art. *Neurocomputing*, 473:37–53.

Huan Ma, Changqing Zhang, Yatao Bian, Lemao Liu, Zhirui Zhang, Peilin Zhao, Shu Zhang, Huazhu Fu, Qinghua Hu, and Bingzhe Wu. 2023. Fairness-guided few-shot prompting for large language models. *Advances in Neural Information Processing Systems*, 36:43136–43155.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.

Timothee Mickus, Elaine Zosa, Raúl Vázquez, Teemu Vahtola, Jörg Tiedemann, Vincent Segonne, Alessandro Raganato, and Marianna Apidianaki. 2024. Semeval-2024 task 6: Shroom, a shared-task on hallucinations and related observable overgeneration mistakes. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1979–1993.

Gabrijela Perković, Antun Drobnjak, and Ivica Botički. 2024. Hallucinations in llms: Understanding and addressing challenges. In *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, pages 2084–2088. IEEE.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.

Hannah Sansford, Nicholas Richardson, Hermina Petric Maretic, and Juba Nait Saada. 2024. Grapheval: A knowledge-graph based llm hallucination evaluation framework. *arXiv preprint arXiv:2407.10793*.

Yongliang Shen, Zeqi Tan, Shuhui Wu, Wenqi Zhang, Rongsheng Zhang, Yadong Xi, Weiming Lu, and Yueting Zhuang. 2023. Promptner: Prompt locating and typing for named entity recognition. *arXiv preprint arXiv:2305.17104*.

Bosheng Song, Fen Li, Yuansheng Liu, and Xiangxiang Zeng. 2021. Deep learning methods for biomedical named entity recognition: a survey and qualitative comparison. *Briefings in Bioinformatics*, 22(6):bbab282.

Charles Spearman. 1987. The proof and measurement of association between two things. *The American journal of psychology*, 100(3/4):441–471.

Weihang Su, Changyue Wang, Qingyao Ai, Yiran Hu, Zhijing Wu, Yujia Zhou, and Yiqun Liu. 2024. Unsupervised real-time hallucination detection based on the internal states of large language models. *arXiv preprint arXiv:2403.06448*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Raúl Vázquez, Timothee Mickus, Elaine Zosa, Teemu Vahtola, Jörg Tiedemann, Aman Sinha, Vincent Segonne, Fernando Sánchez-Vega, Alessandro Raganato, Jindřich Libovický, Jussi Karlgren, Shaoxiong Ji, Jindřich Helcl, Liane Guillou, Ona de Gibert, Jaione Bengoetxea, Joseph Attieh, and Marianna Apidianaki. 2025. SemEval-2025 Task 3: Mu-SHROOM, the multilingual shared-task on hallucinations and related observable overgeneration mistakes.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

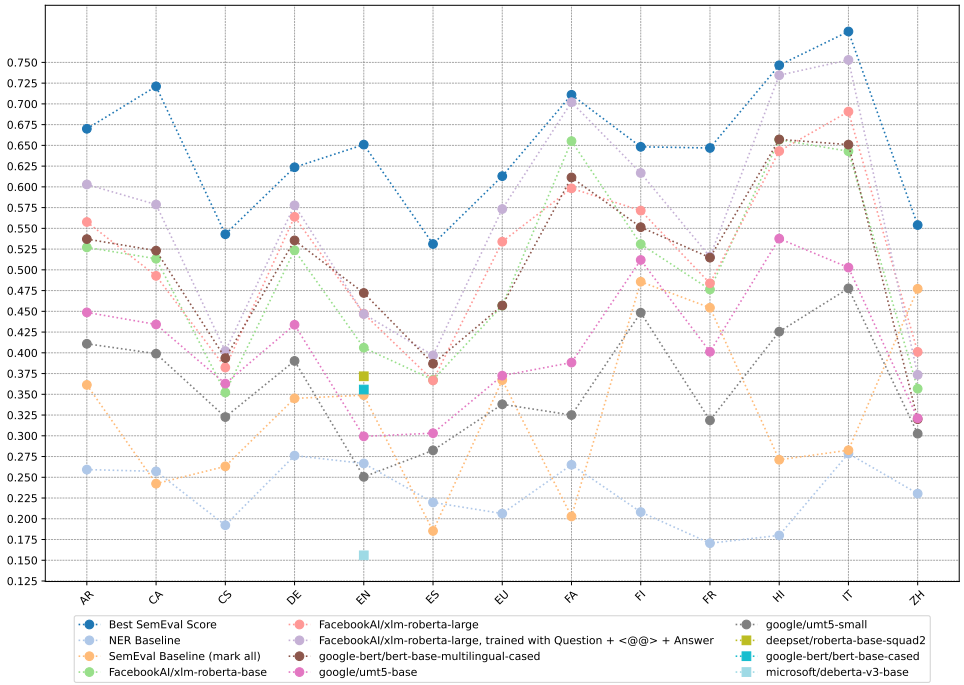
Vikas Yadav and Steven Bethard. 2019. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*.

Xiaoying Zhang, Baolin Peng, Ye Tian, Jingyan Zhou, Lifeng Jin, Linfeng Song, Haitao Mi, and Helen Meng. 2024. Self-alignment for factuality: Mitigating hallucinations in llms via self-evaluation. *arXiv preprint arXiv:2402.09267*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

A IoU Scores for models trained on synthetic data and fine-tuned further with either: (1) all the validation sets (multilingual models); (2) single validation set (English language models)



B Base Models Used

Model	Description
`google-bert/bert-base-cased`	109M parameters, pretrained for masked language modeling (MLM) (Devlin et al., 2018).
`deepset/roberta-base-squad2`	124M parameters, a RoBERTa-base model fine-tuned for extractive question answering (deepset, 2020).
`microsoft/deberta-v3-base`	86M backbone parameters, pretrained for replaced token detection (He et al., 2021).

Table 1: Monolingual (English) base models for fine-tuning

Model	Description
`google-bert/bert-base-multilingual-cased`	179M parameters, 104 languages, pretrained for MLM and next sentence prediction (NSP) (Devlin et al., 2018)
`google/umt5-small`	179M parameters, 102 languages (Chung et al., 2023)
`google/umt5-base`	
`FacebookAI/xlm-roberta-base`	279M (base) / 561M (large) parameters, 94 languages, pretrained for MLM (Conneau et al., 2019a)
`FacebookAI/xlm-roberta-large`	

Table 2: Multilingual base models for fine-tuning

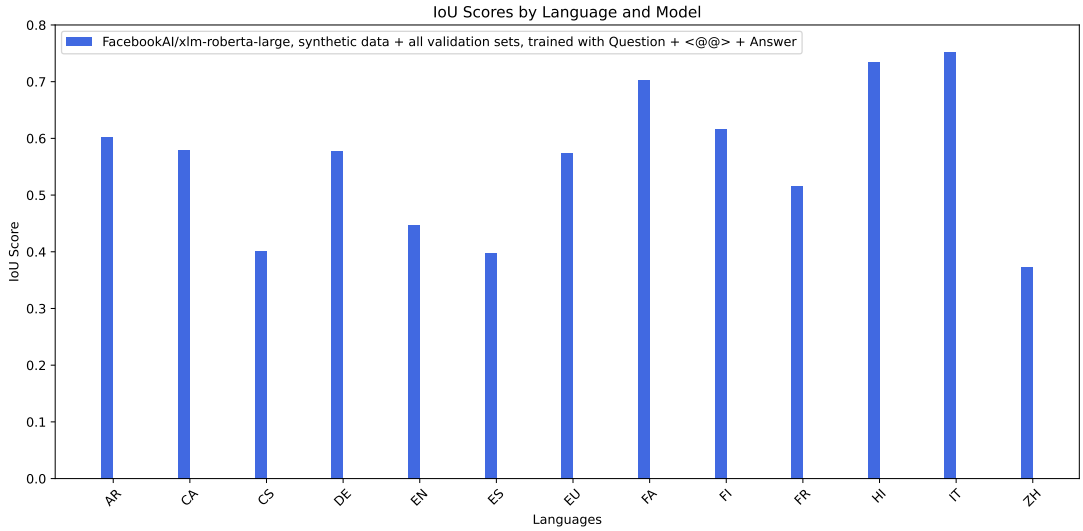
C Prompt for Data Construction

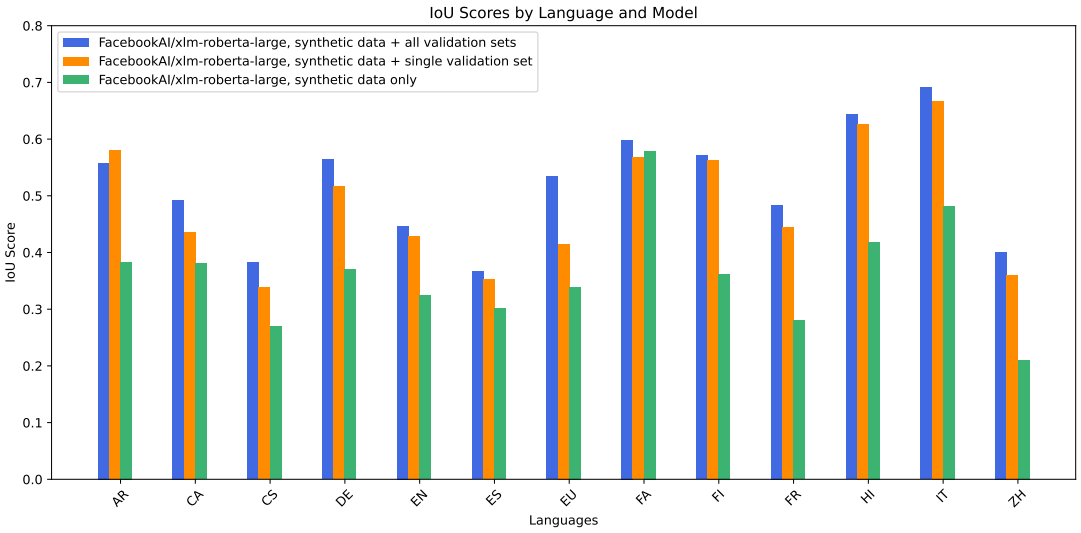
```
""Given information 1:
   "model_input": "How many genera does the
Erysiphales order contain?",
   "model_output_text": "The Elysiphale order
contains 5 genera.",
   "Model_output_text" is supposed to be the
answer to the question in "model_input" but
contains some errors (hallucinations) which
contradict the fact. Your task is to specify the
spans of the erroneous texts and mark them as
"soft_labels".
   In order to do this, firstly, you should
locate the erroneous texts based on the question
asked in "model_input" as well as the fact. For
example, in this case, Elysiphale order contains
4 genera instead of 5; the span of "4" is [31,
32]; and it has a probability of 1 to be
erroneous. Therefore, you should specify this in
the following way:
     { "start": 30, "prob": 1, "end": 31 }.
   Apply the same approach to other parts of the
text. The final result is:
     "soft_labels": [
       { "start": 4, "prob": 0.2, "end": 14 },
       { "start": 30, "prob": 1, "end": 31 },
       { "start": 31, "prob": 0.2, "end": 38 }],
     "hard_labels": [[30, 31]],
For your information, here are two more examples:
Example 1
   "model_input": "Do all arthropods have
antennae?",
   "model_output_text": "Yes, all insects and
arachnids (including spiders, scorpions, and
ticks) have at least one antenna.",
   "soft_labels": [
     { "start": 10, "prob": 0.2, "end": 12 },
     { "start": 12, "prob": 0.3, "end": 13 },
     { "start": 13, "prob": 0.2, "end": 18 },
     { "start": 25, "prob": 0.9, "end": 31 },
     { "start": 31, "prob": 0.1, "end": 37 },
     { "start": 45, "prob": 1, "end": 49 },
     { "start": 49, "prob": 0.3, "end": 65 },
     { "start": 65, "prob": 0.2, "end": 69 },
     { "start": 69, "prob": 0.9, "end": 83 }],
Example 2
   "model_input": "What did Petra van Staveren
win a gold medal for?",
   "model_output_text": "Petra van Stoveren won a
silver medal in the 2008 Summer Olympics in
Beijing, China.",
   "soft_labels": [
     { "start": 10, "prob": 0.2, "end": 12 },
     { "start": 12, "prob": 0.3, "end": 13 },
     { "start": 13, "prob": 0.2, "end": 18 },
     { "start": 25, "prob": 0.9, "end": 31 },
     { "start": 31, "prob": 0.1, "end": 37 },
     { "start": 45, "prob": 1, "end": 49 },
     { "start": 49, "prob": 0.3, "end": 65 },
     { "start": 65, "prob": 0.2, "end": 69 },
     { "start": 69, "prob": 0.9, "end": 83 }],
   "hard_labels": [ [25, 31], [45, 49], [69, 83]
],
You should:
1. Study the examples above, understand why and
how certain texts in "model_output_text" are
labeled.
2. Please generate a similar example, in which
you ask a question, answer it with one or a few
hallucinations deliberately, and label the
hallucinated words (instead of the spans and
probabilities of possible hallucinations).
3. You are encouraged to include more than two
hallucinated words in your output.
4. You do not need to explain your annotations.
5. The output format should be JSON.
The format should be:
   "model_input":
   "model_output_text":
   "hallucinated_words": <list all hallucinated
words in "model_output_text" here>
generate your response in {lang}""
```

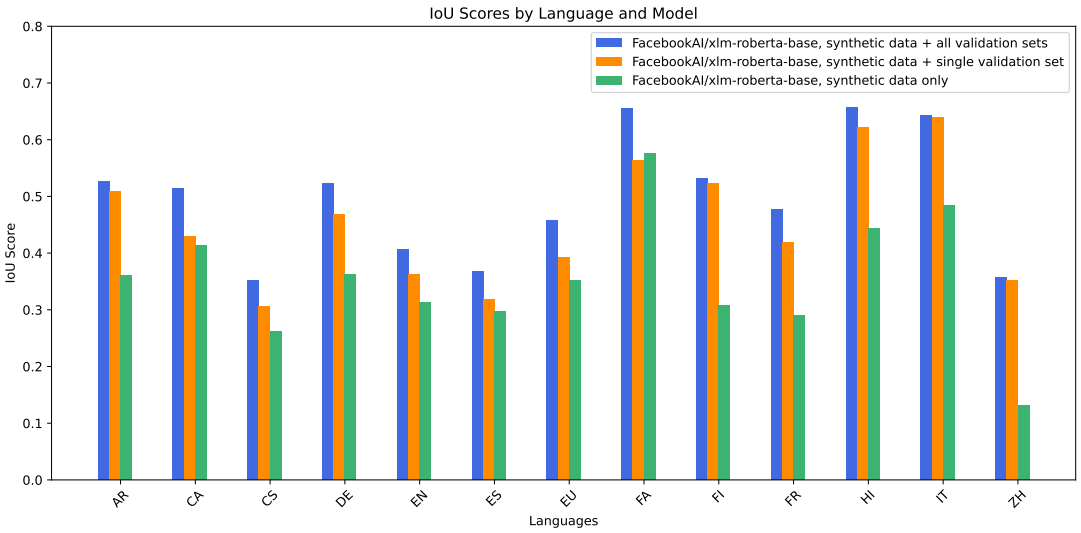
The above prompt was used as an input for dialogue systems, with a language specified inside `{lang}`. The output was then manually copied and pasted into a .jsonl file, repetitive outputs were eliminated. Hard labels were created by running the following example code:

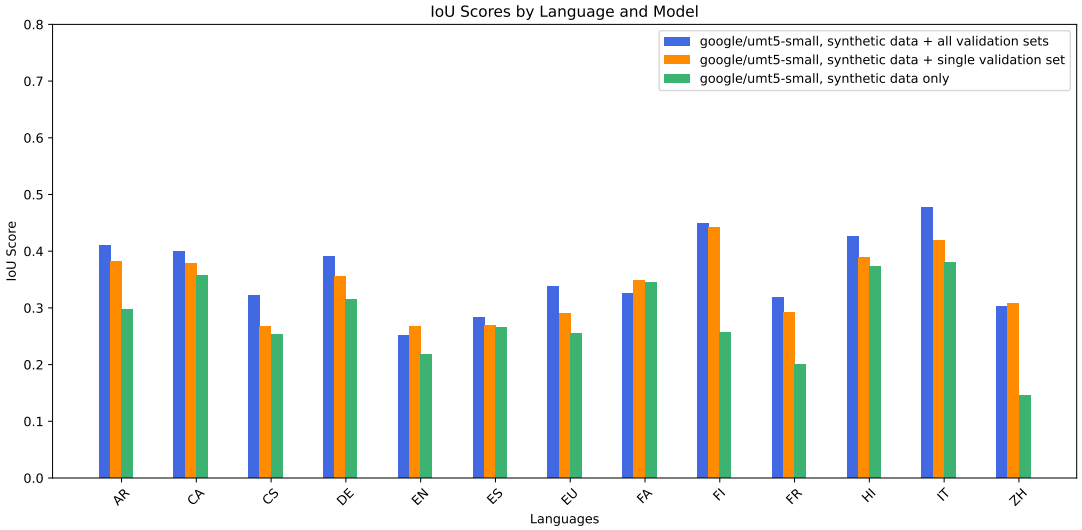
```python
import json

current_id = 1

with open('eu-sim-val-raw.json', 'r', encoding='utf
    -8') as file:
    data = json.load(file)

with open('eu-sim-val.jsonl', 'a', encoding='utf-8')
     as output_file:

    for entry in data:
        model_input = entry["model_input"]
        model_output_text = entry["model_output_text
            "]
        spans = []

        for word in entry["hallucinated_words"]:
            start_index = 0
            while True:
                start_index = model_output_text.find
                    (word, start_index)
                if start_index == -1:
                    break
                end_index = start_index + len(word)
                spans.append([start_index, end_index
                    ])
                start_index = end_index

        result = {
            "id": current_id,
            "created_with": "GPT4o",
            "lang": "eu",
            "model_input": model_input,
            "model_output_text": model_output_text,
            "hard_labels": [[start_index, end_index]
                for start_index, end_index in
                spans]
        }

        current_id += 1

        json.dump(result, output_file, ensure_ascii=
            False)
        output_file.write("\n")
```

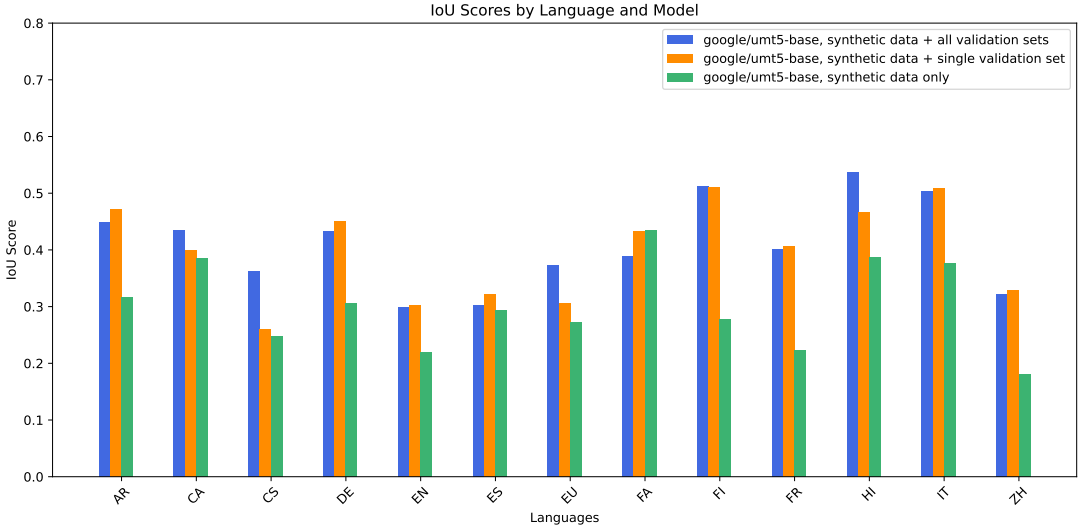
D Comparison of IoU Scores for Different Fine-Tuning Strategies of Multilingual Models

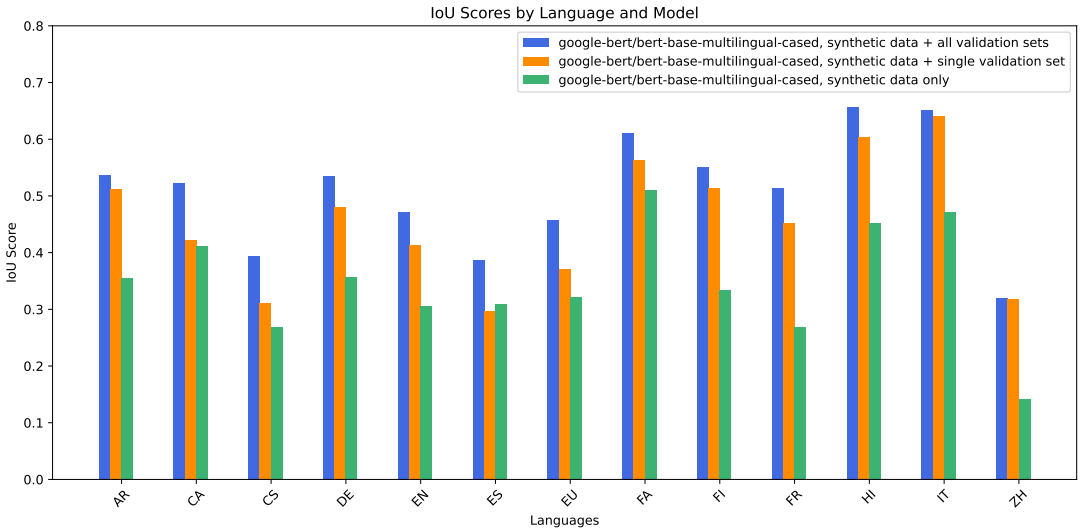







IoU Scores by Language and Model
google/umt5-small, synthetic data + all validation sets
google/umt5-small, synthetic data + single validation set
google/umt5-small, synthetic data only
IoU Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
AR
CA
CS
DE
EN
ES
EU
FA
FI
FR
HI
IT
ZH
Languages


IoU Scores by Language and Model
google/umt5-base, synthetic data + all validation sets
google/umt5-base, synthetic data + single validation set
google/umt5-base, synthetic data only
IoU Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
AR
CA
CS
DE
EN
ES
EU
FA
FI
FR
HI
IT
ZH
Languages


IoU Scores by Language and Model
google-bert/bert-base-multilingual-cased, synthetic data + all validation sets
google-bert/bert-base-multilingual-cased, synthetic data + single validation set
google-bert/bert-base-multilingual-cased, synthetic data only
IoU Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
AR
CA
CS
DE
EN
ES
EU
FA
FI
FR
HI
IT
ZH
Languages

E IoU Scores for All the Models

Multilingual Models	AR	CA	CS	DE	EN	ES	EU	FA	FI	FR	HI	IT	ZH
FacebookAI/xlm-roberta-base, synth. data + all val. sets	0.527	0.513	0.352	0.524	0.406	0.367	0.457	0.655	0.531	0.476	0.657	0.643	0.357
FacebookAI/xlm-roberta-base, synth. data + 1 val. set	0.509	0.430	0.305	0.468	0.363	0.319	0.392	0.564	0.523	0.420	0.622	0.638	0.352
FacebookAI/xlm-roberta-base, synth. data only	0.361	0.414	0.262	0.363	0.313	0.298	0.351	0.576	0.308	0.290	0.444	0.485	0.131
FacebookAI/xlm-roberta-large, synth. data + all val. sets	0.558	0.493	0.382	0.564	0.447	0.367	0.534	0.598	0.571	0.484	0.643	0.691	0.401
FacebookAI/xlm-roberta-large, synth. data + 1 val. set	0.580	0.436	0.338	0.516	0.429	0.353	0.415	0.567	0.563	0.445	0.627	0.667	0.361
FacebookAI/xlm-roberta-large, synth. data only	0.383	0.380	0.269	0.370	0.324	0.301	0.339	0.579	0.362	0.280	0.419	0.482	0.211
FacebookAI/xlm-roberta-large, trained with QA pairs	0.603	0.579	0.402	0.578	0.447	0.397	0.573	0.702	0.617	0.516	0.735	0.753	0.373
google-bert/bert-base-multilingual-cased, synth. data + all val. sets	0.537	0.523	0.394	0.535	0.472	0.387	0.457	0.611	0.551	0.515	0.657	0.651	0.320
google-bert/bert-base-multilingual-cased, synth. data + 1 val. set	0.512	0.422	0.311	0.480	0.414	0.297	0.371	0.563	0.513	0.451	0.604	0.641	0.318
google-bert/bert-base-multilingual-cased, synth. data only	0.355	0.412	0.268	0.356	0.305	0.310	0.322	0.511	0.334	0.268	0.452	0.471	0.142
google/umt5-base, synth. data + all val. sets	0.449	0.434	0.363	0.434	0.299	0.303	0.372	0.388	0.512	0.401	0.538	0.503	0.321
google/umt5-base, synth. data + 1 val. set	0.471	0.399	0.260	0.451	0.302	0.322	0.306	0.433	0.510	0.406	0.467	0.509	0.329
google/umt5-base, synth. data only	0.317	0.385	0.249	0.307	0.220	0.293	0.273	0.434	0.278	0.224	0.387	0.376	0.181
google/umt5-small, synth. data + all val. sets	0.411	0.399	0.323	0.390	0.250	0.282	0.338	0.325	0.448	0.319	0.425	0.478	0.303
google/umt5-small, synth. data + 1 val. set	0.381	0.378	0.267	0.355	0.268	0.269	0.290	0.348	0.442	0.291	0.389	0.420	0.308
google/umt5-small, synth. data only	0.298	0.358	0.252	0.315	0.217	0.265	0.255	0.345	0.257	0.200	0.373	0.380	0.146
Monolingual Models													
deepset/roberta-base-squad2, synth. data + 1 val. set					0.372								
deepset/roberta-base-squad2, synth. data only					0.333								
google-bert/bert-base-cased, synth. data + 1 val. set					0.356								
google-bert/bert-base-cased, synth. data only					0.376								
microsoft/deberta-v3-base, synth. data + 1 val. set					0.156								
microsoft/deberta-v3-base, synth. data only					0.145								