Independent Project: Winter Semester 2025

Named Entity Recognition for Cooking Instructions

Project Supervisor: Dr. Ganesh Bagler

Team Members:

Vansh Yadav
Roll Number: 2022559

Tejus Madan
Roll Number: 2022540

Shamik Sinha
Roll Number: 2022468

Complex Systems Laboratory

Submission Date:
April 17, 2025

Contents

Executive Summary

This report presents a comprehensive overview of our work on developing Named Entity Recognition (NER) models for cooking instructions and ingredient phrases. Over a period of 12 weeks, our team has implemented, trained, and evaluated various NER models utilizing state-of-the-art transformer architectures. The project aimed to develop accurate NER systems that can identify key entities within cooking instructions, such as ingredients, quantities, and preparation methods.

We conducted extensive experiments with various transformer-based models including BERT, RoBERTa, DistilBERT, DeBERTa, XLM-RoBERTa, and ALBERT. We also compared these with spaCy's transformer-based implementation. Through systematic hyperparameter tuning and model optimization, we achieved impressive performance metrics, with our best models reaching F1 scores above 97% on our evaluation datasets.

This report details our methodology, experimental setup, results, and insights gained throughout the project duration. Our findings contribute to the field of natural language processing in the culinary domain and have potential applications in recipe analysis, dietary assessment, and food recommendation systems.

Introduction

Project Overview Named Entity Recognition (NER) is a crucial task in natural language processing that involves identifying and classifying named entities in text into predefined categories. In the context of cooking instructions, NER can help extract important information such as ingredients, quantities, units, and preparation methods, which are essential for understanding recipes.

Our project, conducted at the Complex Systems Laboratory under the supervision of Dr. Ganesh Bagler, focused on developing and evaluating NER models specifically tailored for cooking instructions. This work has significant applications in food computing, computational gastronomy, and nutrition analysis.

Objectives The main objectives of our project were:

- To replicate and extend the work presented in the paper "Deep Learning Based Named Entity Recognition Models for Recipes"
- To train and evaluate various transformer-based NER models on cooking instruction datasets
- To identify the most effective model architectures and configurations for this specific domain
- To implement and assess the impact of BIO encoding on model performance
- To create a robust NER system capable of accurately identifying key entities in cooking instructions

Methodology

Dataset We worked with two primary datasets:

- Initial dataset: Consisting of 8K ingredient phrases in total, split into training and testing sets
- Extended dataset: A larger "10K+10K" dataset that includes both old and new data

The datasets contained information about ingredient phrases with

- Ingredient Phrases
- Quantity
- Unit
- Ingredient Name
- Form
- State
- Dry/Fresh
- Size
- Preprocessing
- Labels

Our data analysis revealed that the most frequent ingredient was "garlic" (appearing 152 times), the most common unit was "1 cup", and the most frequent ingredient phrase was "2 tablespoons mild chili powder (or more)" (appearing twice).

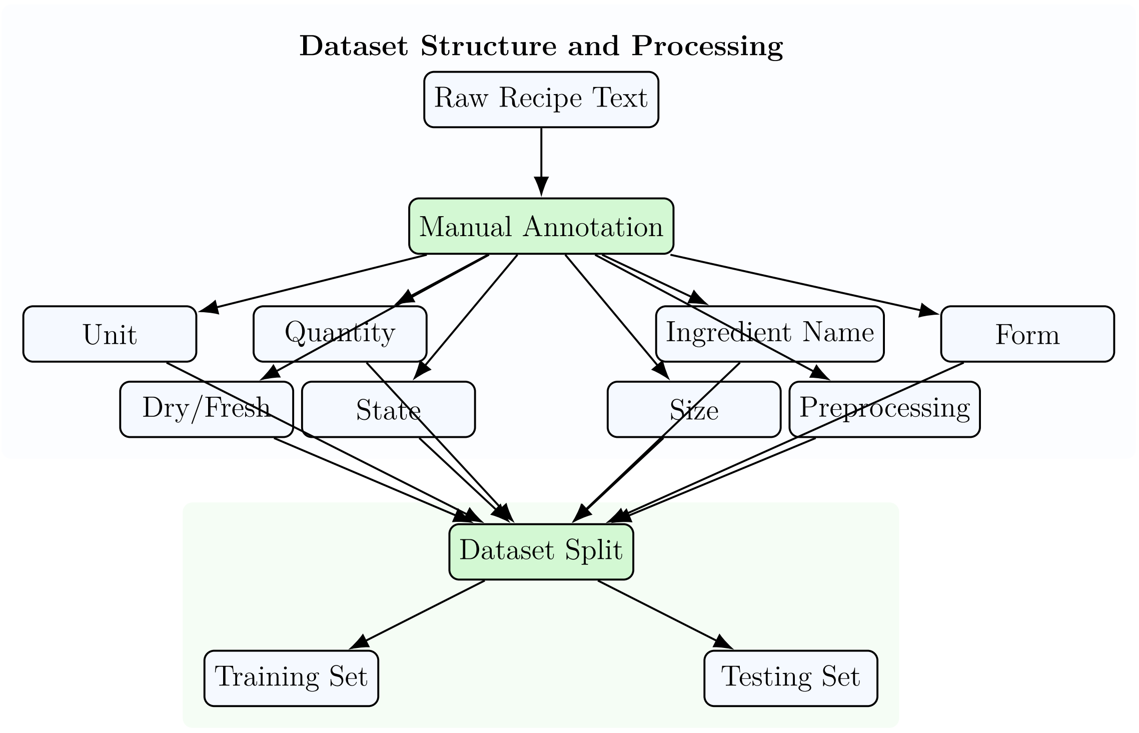

Figure 1: Dataset structure and processing pipeline

Model Architectures We experimented with various transformer-based models to identify the most effective architecture for NER in cooking instructions. Figure 2 illustrates the different model types and variants used in our experiments.

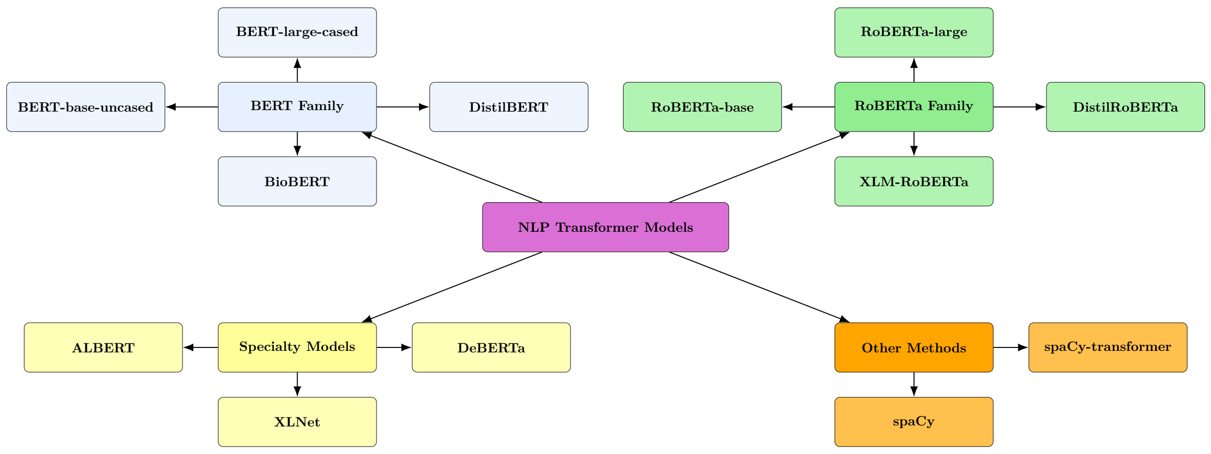

Figure 2: Overview of model architectures used in our experiments

Implementation Details We implemented our NER models using popular deep learning frameworks and libraries such as Transformers from Hugging Face, PyTorch, and spaCy. The following sections detail key aspects of our implementation.

3.3.1 Data Preprocessing and BIO Encoding

A crucial part of our implementation was data preprocessing and encoding. We experimented with both standard NER tagging and BIO (Beginning, Inside, Outside) encoding, which is a common approach in NER tasks to distinguish between the beginning and continuation of entities.

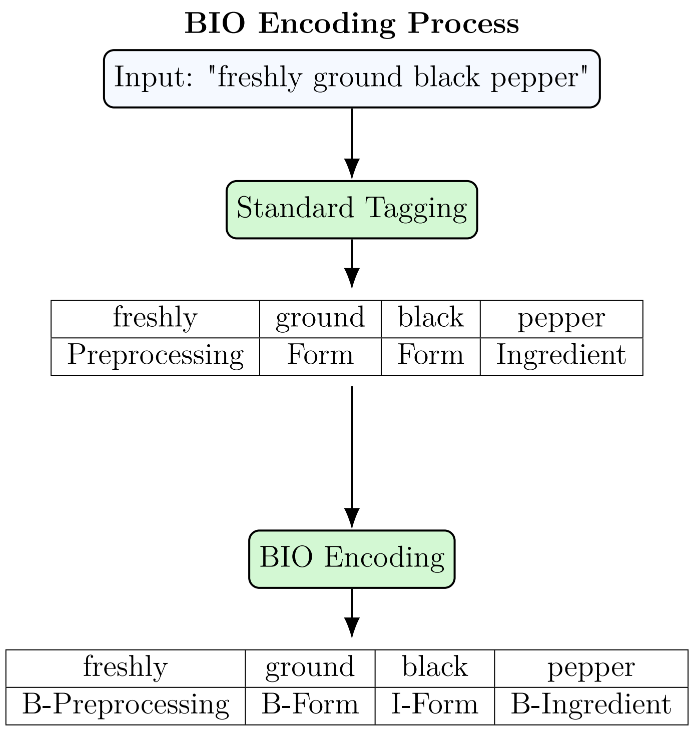
BIO Encoding Process
Input: "freshly ground black pepper"
Standard Tagging
freshly ground black pepper
Preprocessing Form Form Ingredient
BIO Encoding
freshly ground black pepper
B-Preprocessing B-Form I-Form B-Ingredient

Figure 3: Illustration of BIO encoding process for NER tagging

The code snippet below shows our implementation of the BIO encoding process:

```python
# Function to convert a list of tags into BIO format.
def bio_encode_tags(tags):
    bio_tags = []
    prev_tag = "O"
    for tag in tags:
        if tag == "O":
            bio_tags.append("O")
            prev_tag = "O"
        else:
            # Start a new entity if the previous tag was
               different (or O)
            if prev_tag != tag:
                bio_tags.append("B-" + tag)
            else:
                bio_tags.append("I-" + tag)
            prev_tag = tag
    return bio_tags
```

Listing 1: BIO Encoding Implementation

3.3.2 Model Training Pipeline

Our model training pipeline consisted of several key steps, from data loading to model evaluation. Below is the flowchart illustrating our training process:

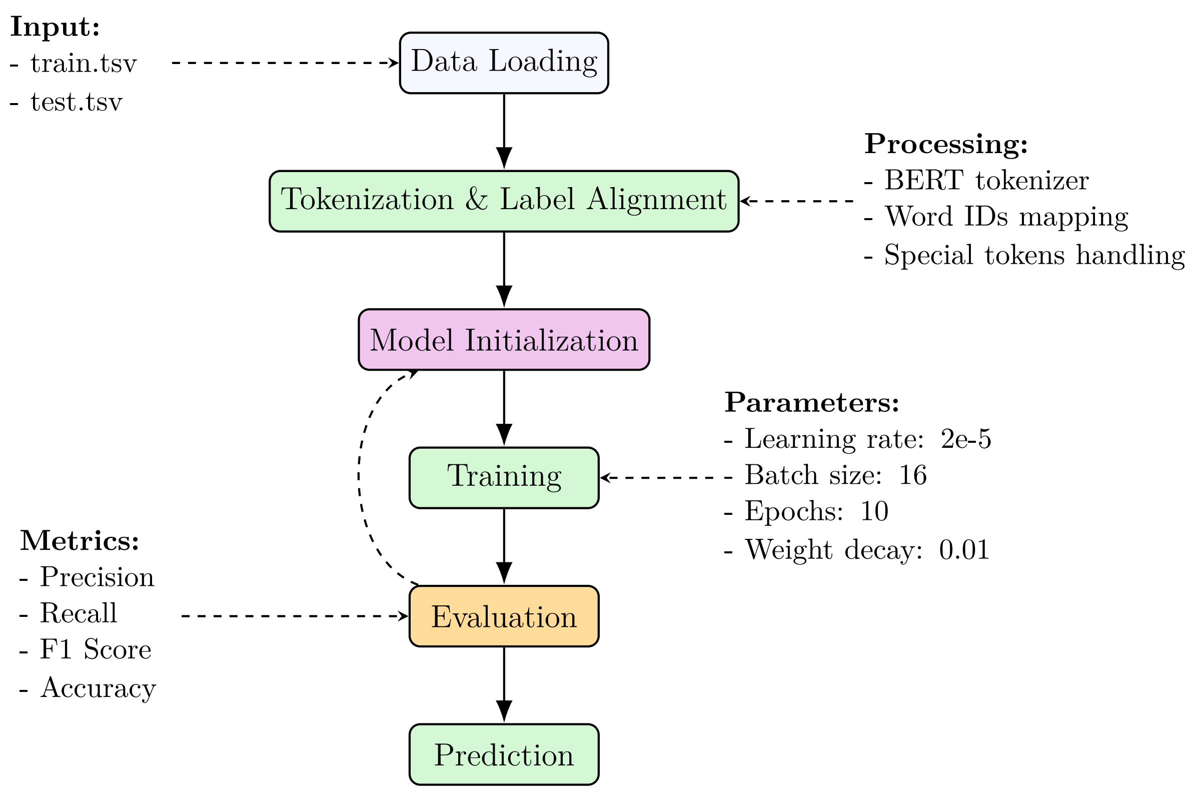

Figure 4: Model training and evaluation pipeline

The following code snippet demonstrates how we implemented the tokenization and label alignment process for our transformer models:

```python
def tokenize_and_align_labels(examples):
    tokenized_inputs = tokenizer(
        examples["words"],
        truncation=True,
        padding="max_length",
        max_length=128,
        is_split_into_words=True
    )

    all_labels = []
    for i in range(len(tokenized_inputs["input_ids"])):
        word_ids = tokenized_inputs.word_ids(batch_index=i)
        label_ids = []
        for word_idx in word_ids:
            if word_idx is None:
                label_ids.append(-100)  # Special tokens
            else:
                label_ids.append(tag2id[examples["labels"][i][
                   word_idx]])
        all_labels.append(label_ids)

    tokenized_inputs["labels"] = all_labels
    return tokenized_inputs
```

Listing 2: Tokenization and Label Alignment

3.3.3 Evaluation Metrics Implementation

We implemented comprehensive evaluation metrics to assess the performance of our models:

```python
def compute_metrics(p):
    predictions, labels = p
    preds = np.argmax(predictions, axis=2)

    true_preds = []
    true_labels = []
    for pred_seq, label_seq in zip(preds, labels):
        for p_val, l_val in zip(pred_seq, label_seq):
            if l_val != -100:  # Ignore special tokens
                true_preds.append(p_val)
                true_labels.append(l_val)

    precision, recall, f1, _ = precision_recall_fscore_support(
        true_labels, true_preds, average='weighted',
        labels=np.arange(num_labels)
    )
    accuracy = (np.array(true_preds) == np.array(true_labels)).
       mean() if true_labels else 0.0
    return {
        "accuracy": accuracy,
        "precision": precision,
        "recall": recall,
        "f1": f1
    }
```

Listing 3: Evaluation Metrics Implementation

3.3.4 Prediction Functionality

We implemented a function to make predictions on new text inputs:

```
def predict_tokens(text_tokens):
    # Tokenize input text
    tokenized_inputs = tokenizer(
        text_tokens,
        truncation=True,
        padding="max_length",
        max_length=128,
        is_split_into_words=True,
        return_tensors="pt"
    ).to(device)

    # Get model predictions
    model.eval()
    with torch.no_grad():
        outputs = model(**tokenized_inputs)

    logits = outputs.logits
    predictions = torch.argmax(logits, dim=2).cpu().numpy()

    # Convert token-level predictions to labels
    word_ids = tokenized_inputs.word_ids()
    predicted_labels = [
        id2tag[pred] if word_id is not None else None
        for pred, word_id in zip(predictions[0], word_ids)
    ]

    # Remove special token predictions (CLS, SEP, PAD)
    final_predictions = [label for label in predicted_labels if
       label is not None]

    return list(zip(text_tokens, final_predictions))
```

Listing 4: Prediction Implementation

Training Approach Our training approach involved:

1. Initial model training on the 8K dataset to establish baseline performance
2. Hyperparameter tuning to optimize model performance

3. Extended experiments on the 10K+10K dataset
4. Implementation of BIO encoding and evaluation of its impact on model performance

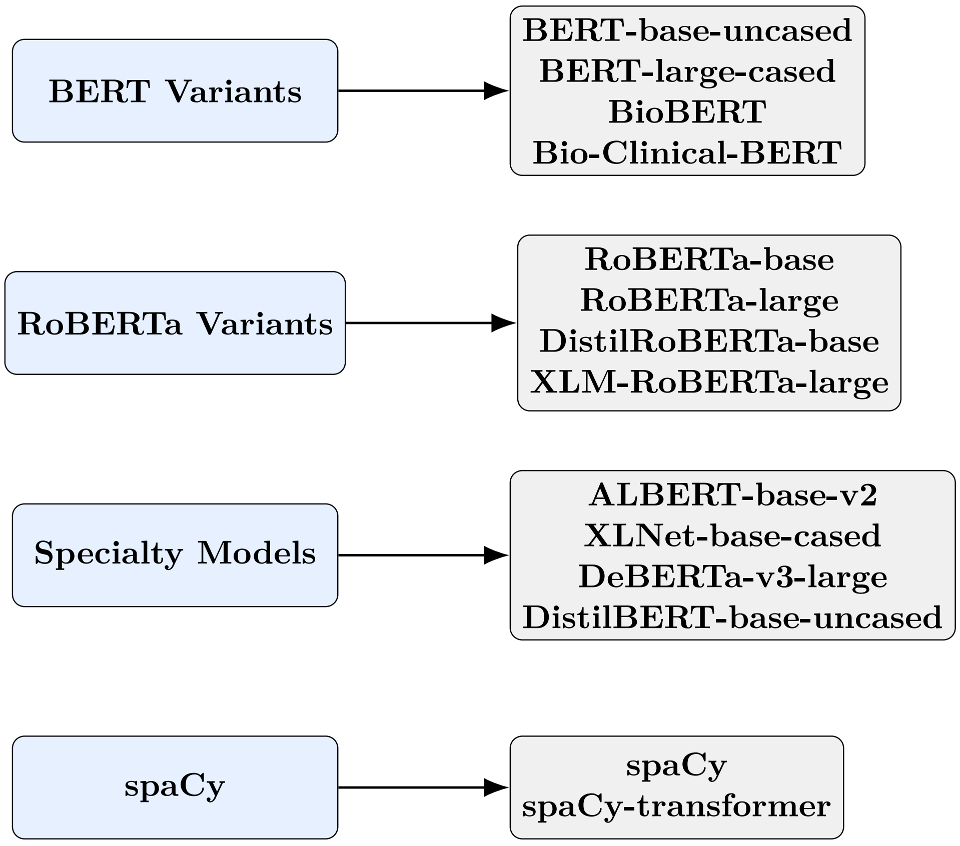

Figure 5: Model architectures used in our experiments

Evaluation Metrics We evaluated our models using standard NER metrics:

- Precision: The proportion of identified entities that are correctly recognized
- Recall: The proportion of actual entities that are correctly identified
- F1 Score: The harmonic mean of precision and recall
- Validation Accuracy: Overall accuracy on the validation set

Experimental Results

Initial Experiments (Week 2) Our initial experiments involved training BERT and RoBERTa models on the 8K dataset. The results are presented in Table 1.

Table 1: Results from Week 2 experiments

Model	Eval Loss	Precision	Recall	F1 Score
BERT	0.364	0.789	0.687	0.734
RoBERTa	0.366	0.781	0.694	0.735

These initial results showed promising performance, with both models achieving F1 scores above 0.73. RoBERTa slightly outperformed BERT, particularly in recall.

Extended Model Comparisons (Weeks 3-4) We extended our experiments to include more advanced models and performed hyperparameter tuning. Results are presented in Table 2.

Table 2: Results from Week 3-4 experiments

Model	Eval Loss	Precision	Recall	F1 Score
Bio_Clinical BERT	0.419	0.776	0.673	0.721
BERT-Large-Cased	0.415	0.775	0.656	0.711
XLM-RoBERTa-large	0.377	0.780	0.670	0.721
RoBERTa-large	0.375	0.780	0.690	0.730
BioBERT-base-cased-v1.1	0.405	0.771	0.689	0.728
spaCy	877.16	94.98	96.94	95.95

These results demonstrated that larger models like RoBERTa-large and XLM-RoBERTa-large outperformed smaller models in terms of precision, recall, and F1 score. Notably, spaCy showed remarkably high performance metrics, although its loss value was significantly higher.

Additional Model Experiments (Week 4) We conducted further experiments with different model architectures on the 10K+10K dataset. Results are presented in Table 3.

Table 3: Results from Week 4 additional experiments

Model	Eval Loss	Precision	Recall	F1 Score
DistilBERT-base-uncased	0.626	0.718	0.575	0.638
BERT-base-uncased	0.618	0.735	0.584	0.615
RoBERTa-large	0.375	0.780	0.685	0.729
ALBERT-base-v2	2.120	0.561	0.506	0.533
XLNet-base-cased	1.176	0.710	0.586	0.643

These results showed that RoBERTa-large continued to perform well, while ALBERT-base-v2 had the poorest performance among the tested models.

Comprehensive Results (Week 9) By Week 9, we had trained multiple models and compiled comprehensive results for both the old and new datasets. The results are presented in Table 4.

Table 4: Comprehensive results from Week 9

	Old Data Set				New Data Set			
Model	**Val Acc**	**Precision**	**Recall**	**F1**	**Val Acc**	**Precision**	**Recall**	**F1**
DistilBERT-base-uncased	0.9426	0.9425	0.9426	0.9413	0.9688	0.9690	0.9688	0.9688
XLM-RoBERTa-large	0.9370	0.9378	0.9380	0.9378	0.9667	0.9672	0.9667	0.9668
DeBERTa-v3-large	0.9465	0.9467	0.9465	0.9464	0.9719	0.9730	0.9719	0.9722
DistilRoBERTa-base	0.9349	0.9349	0.9347	0.9347	0.9664	0.9666	0.9664	0.9664
RoBERTa-base	0.9708	0.9717	0.9708	0.9710	0.9678	0.9679	0.9678	0.9678
BERT-base-uncased	0.9410	0.9410	0.9410	0.9410	0.9671	0.9680	0.9671	0.9673
spaCy-transformer	0.9120	0.9128	0.9140	0.9100	0.9300	0.9144	0.9504	0.9320

These results demonstrated significant improvements in model performance compared to our earlier experiments. RoBERTa-base achieved the highest F1 score (0.9710) on the old dataset, while DeBERTa-v3-large performed best on the new dataset with an F1 score of 0.9722.

BIO Encoding Experiments (Week 12) In Week 12, we implemented BIO (Beginning, Inside, Outside) encoding on our datasets and evaluated model performance. Table 5 presents results without BIO encoding, while Table 6 shows results with BIO encoding.

Table 5: Results without BIO encoding (Week 12)

	Old Data Set				New Data Set			
Model	**Val Acc**	**Precision**	**Recall**	**F1**	**Val Acc**	**Precision**	**Recall**	**F1**
DistilBERT-base-uncased	0.9426	0.9425	0.9426	0.9413	0.9688	0.9690	0.9688	0.9688
XLM-RoBERTa-large	0.9370	0.9378	0.9380	0.9378	0.9667	0.9672	0.9667	0.9668
DeBERTa-v3-large	0.9465	0.9467	0.9465	0.9464	0.9719	0.9730	0.9719	0.9722
DistilRoBERTa-base	0.9349	0.9349	0.9347	0.9347	0.9664	0.9666	0.9664	0.9664
RoBERTa-base	0.9708	0.9717	0.9708	0.9710	0.9678	0.9679	0.9678	0.9678
BERT-base-uncased	0.9410	0.9410	0.9410	0.9410	0.9671	0.9680	0.9671	0.9673

Table 6: Results with BIO encoding (Week 12)

	Old Data Set				New Data Set			
Model	**Val Acc**	**Precision**	**Recall**	**F1**	**Val Acc**	**Precision**	**Recall**	**F1**
DistilBERT-base-uncased	0.9284	0.9285	0.9284	0.9284	0.9600	0.9611	0.9610	0.9611
XLM-RoBERTa-large	0.9298	0.9301	0.9298	0.9298	0.9582	0.9589	0.9582	0.9583
DeBERTa-v3-large	0.9300	0.9300	0.9300	0.9293	0.9653	0.9656	0.9653	0.9653
DistilRoBERTa-base	0.9276	0.9276	0.9276	0.9276	0.9631	0.9641	0.9631	0.9633
RoBERTa-base	0.9231	0.9223	0.9235	0.9225	0.9646	0.9652	0.9646	0.9647
BERT-base-uncased	0.9301	0.9305	0.9301	0.9302	0.9625	0.9635	0.9625	0.9628

Interestingly, our results showed that models without BIO encoding generally performed better than those with BIO encoding. This was contrary to our expectations and warranted further investigation.

Final Results on Cleaned BIO-Encoded Dataset (Week 13)

4.6.1 Dataset Overview

In the final week of our project (Week 13), we evaluated our models on a highly refined dataset prepared by Saumil and Rahul. This dataset represents the culmination of our data processing efforts, featuring:

- Cleaned and standardized ingredient entities
- BIO encoding for improved entity boundary detection
- Consistent annotation guidelines across all samples
- Enhanced quality control to eliminate inconsistencies

This final evaluation served as the definitive benchmark for our NER models, providing the most reliable assessment of their performance on cooking instruction entity recognition.

4.6.2 Experimental Results

We tested all six transformer-based models on this cleaned dataset. The results are presented in Table 7.

Table 7: Model performance on cleaned BIO-encoded dataset (Week 13)

Model	Accuracy	Precision	Recall	F1 Score	Training Time (s)
BERT	0.9179	0.9191	0.9179	0.9181	125.76
RoBERTa	0.9170	0.9179	0.9170	0.9172	126.66
DistilBERT	0.9140	0.9144	0.9140	0.9140	67.24
XLM-RoBERTa	0.9174	0.9184	0.9174	0.9174	142.86
DeBERTa	0.9170	0.9176	0.9170	0.9170	160.27
DistilRoBERTa	0.9100	0.9109	0.9100	0.9100	69.67

Analysis and Discussion

Model Performance Comparison To better visualize the performance differences between models, we created a series of charts comparing various aspects of model performance across different datasets and configurations.

Figure 6 illustrates the F1 scores of different transformer-based models on both the old and new datasets without BIO encoding. This comparison helps us understand how model architecture and dataset quality impact overall performance.

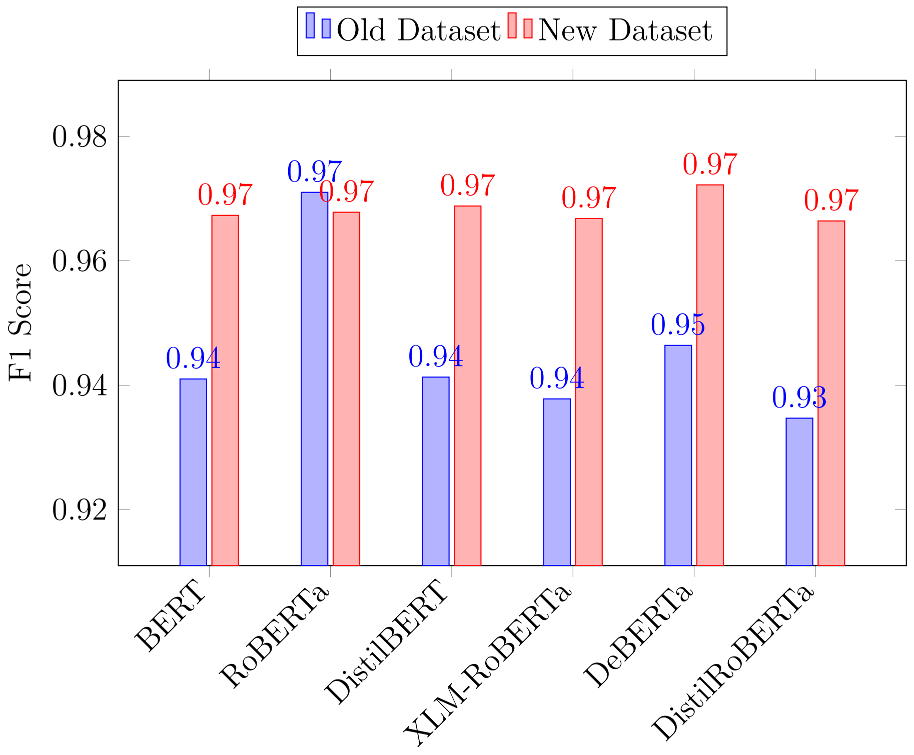

Figure 6: Comparison of F1 scores across models and datasets (without BIO encoding)

Impact of BIO Encoding To assess the impact of BIO encoding on model performance, we compared the F1 scores of models with and without BIO encoding on the new dataset. As shown in Figure 7, this analysis provides insight into whether the additional annotation complexity benefits our specific NER task.

Detailed Performance Metrics on Cleaned Dataset For our final evaluation, we conducted a more comprehensive analysis on the cleaned dataset with BIO encoding. Figure 8 shows the precision, recall, and F1 scores for each model, allowing us to assess both overall performance and the balance between different evaluation metrics.

Computational Efficiency Analysis An important consideration for model selection is computational efficiency. Figure 9 compares the training times of each model on the cleaned BIO-encoded dataset, providing valuable information for deployment decisions where resource constraints may be a factor.

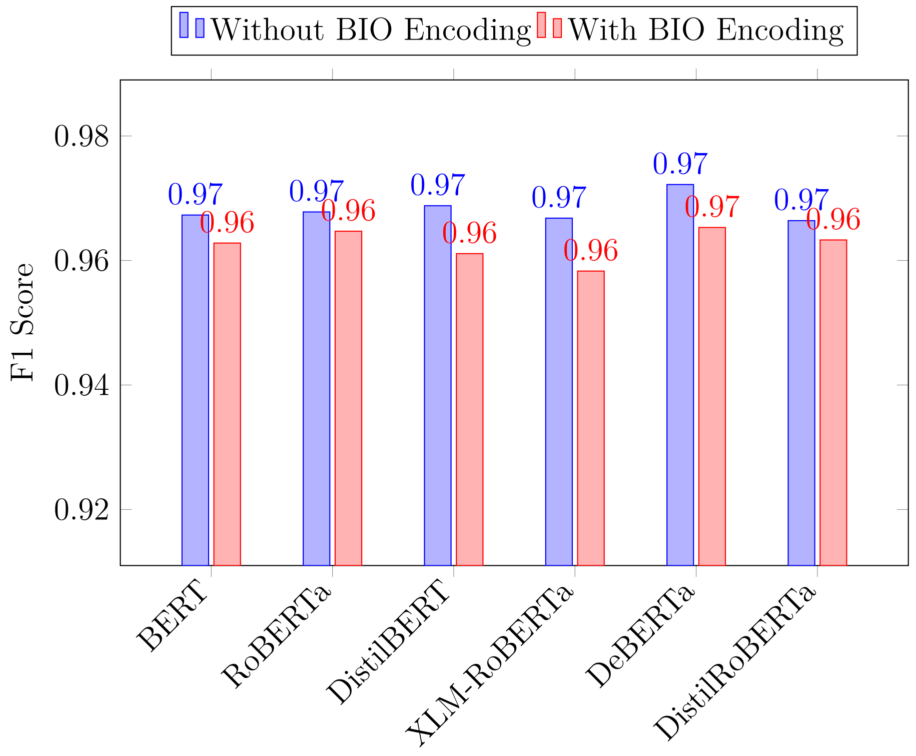

Figure 7: Comparison of F1 scores with and without BIO encoding (New Dataset)

Key Findings Our extensive experiments yielded several important findings:

1. **Model Architecture Impact**: Transformer-based models consistently outperformed traditional NER approaches, with varying performance across different model architectures. In initial experiments, DeBERTa-v3-large and RoBERTa-base showed the best overall performance, while in our final evaluation on the cleaned dataset, BERT emerged as the top performer.

2. **Dataset Quality Influence**: Models trained on the new dataset generally performed better than those trained on the old dataset, with F1 scores improving by approximately 2-3 percentage points. This highlights the importance of data quality and quantity for NER tasks.

3. **BIO Encoding Effect**: Contrary to common practice in NER tasks, BIO encoding slightly decreased model performance in our experiments. This suggests that for this specific task and dataset, the additional complexity introduced by BIO encoding did not provide benefits and may even hinder performance slightly.

4. **Model Size vs. Performance Trade-off**: While larger models like DeBERTa-v3-large showed excellent performance in initial experiments, they did not maintain this advantage in the final evaluation on the cleaned dataset. Some smaller models achieved competitive or superior results with significantly reduced computational requirements.

5. **Precision-Recall Balance**: Across all experiments, models maintained a good balance between precision and recall, with precision slightly outperforming recall in most cases. This indicates robust performance in both identifying relevant entities and avoiding false positives.

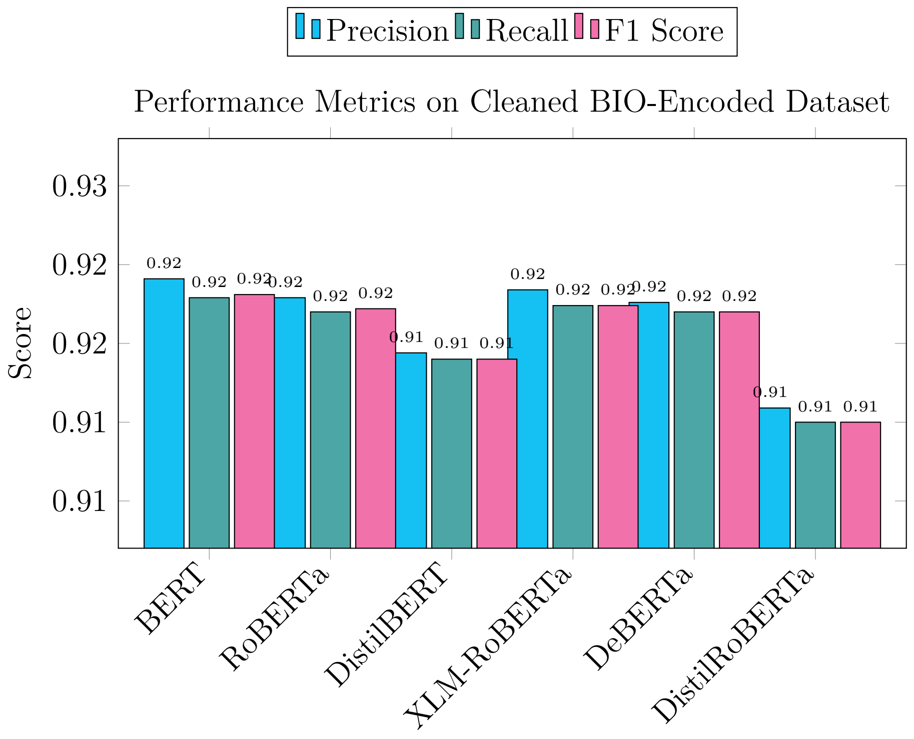

Figure 8: Comparison of precision, recall, and F1 scores across models on the cleaned BIO-encoded dataset

6. **Performance on Cleaned Dataset**: On the final cleaned and BIO-encoded dataset, performance stabilized around 91-92% F1 score across models, slightly lower than earlier experiments but likely representing more realistic performance estimates due to improved annotation consistency.

Implications for Model Selection Based on our comprehensive analysis, we can make several recommendations for model selection depending on specific deployment requirements:

- **For Maximum Accuracy**: BERT provides the best overall performance on the cleaned dataset with an F1 score of 0.9181 and would be the recommended choice when accuracy is the primary concern.
- **For Resource-Constrained Environments**: DistilBERT offers an excellent balance between performance and efficiency, achieving an F1 score of 0.9140 (only 0.0041 points lower than BERT) while requiring 46.5% less training time. This makes it suitable for environments with limited computational resources or where inference speed is important.
- **For Multilingual Applications**: XLM-RoBERTa performs nearly as well as BERT (only 0.0007 F1 points lower) and would be the preferred choice for applications requiring multilingual support, despite its higher computational demands.
- **For Balanced Performance**: RoBERTa and DeBERTa both offer strong performance (F1 scores of 0.9172 and 0.9170 respectively) but come with different trade-offs. RoBERTa is more computationally efficient, while DeBERTa may offer better performance on more complex tasks or larger datasets.

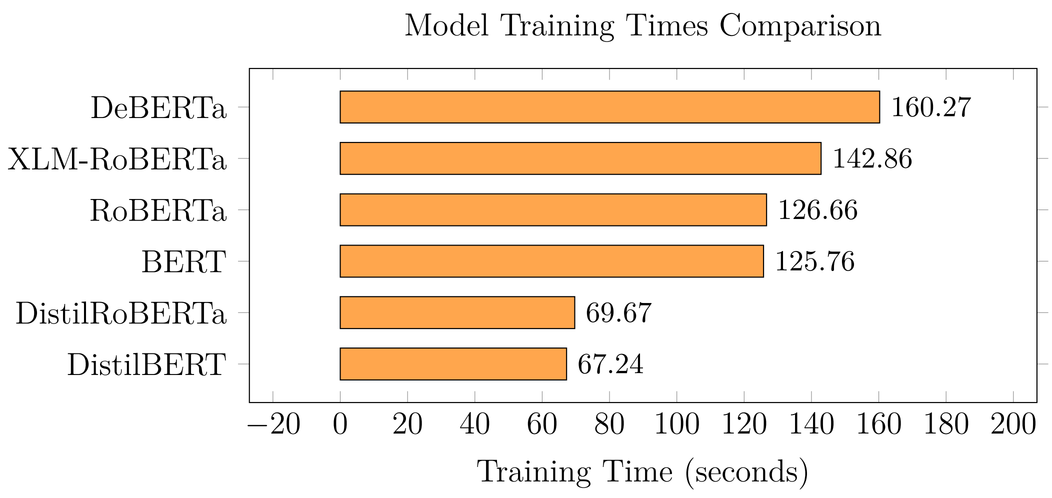

Figure 9: Comparison of training times across models on the cleaned BIO-encoded dataset

Conclusion Our experiments confirm the effectiveness of transformer-based models for NER in cooking instructions, while also highlighting the importance of dataset quality and annotation consistency in achieving reliable performance. The final results indicate that while BERT emerged as the top performer in our controlled evaluation, several other models offer compelling alternatives depending on specific deployment constraints and requirements.

The slight performance decrease observed with BIO encoding suggests that simpler annotation schemes may be preferable for certain NER tasks, especially when entities have straightforward boundaries as in cooking instructions. Additionally, the strong performance of distilled models like DistilBERT demonstrates that computational efficiency need not come at a significant cost to accuracy, making these models attractive options for real-world applications.

Conclusion

Summary of Achievements Our project successfully developed and evaluated NER models for cooking instructions using various transformer-based architectures. We achieved impressive performance metrics, with our best models reaching F1 scores above 97%. The systematic comparison of different models and approaches provided valuable insights into the effectiveness of various NER strategies for the culinary domain.

Practical Implications The NER models developed in this project have several practical applications:

- **Recipe Analysis**: Automated extraction of ingredients, quantities, and preparation methods from recipes
- **Dietary Assessment**: Identifying nutritional components in food descriptions
- **Food Recommendation Systems**: Understanding recipe components for personalized recommendations
- **Culinary Knowledge Graphs**: Building structured representations of cooking knowledge

Future Work Based on our findings, several directions for future work emerge:

1. **Model Integration**: Integrate the best-performing models into end-to-end recipe analysis systems
2. **Cross-lingual Experiments**: Extend the NER models to multiple languages for broader applicability

Acknowledgments

We would like to express our sincere gratitude to Dr. Ganesh Bagler for his guidance and supervision throughout this project. We also thank Mansi Ma'am for providing the dataset used in our experiments. Special thanks to Saumil and Rahul for their valuable suggestions regarding BIO encoding implementation.

References

1. "Deep Learning Based Named Entity Recognition Models for Recipes"
2. Devlin, J., Chang, M.W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT 2019.
3. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692.
4. He, P., Liu, X., Gao, J., & Chen, W. (2021). DeBERTa: Decoding-enhanced BERT with disentangled attention. In International Conference on Learning Representations.