Advanced NLP: Take Home Exam

Orlando Closs

1 Introduction

Evaluating Natural Language Processing (NLP) models by measuring accuracy on held-out datasets can lead to misleading estimates compared to their real-world performance (Ribeiro et al., 2020). This happens because training data often includes biases, meaning it doesn't fully reflect the vairety of inputs that models face in real-world scenarios. Therefore, models may perform on these datasets by recognizing specific data patterns rather than genuinely understanding language or developing the ability to generalize. To address this limitation, this report explores behavioral testing: an approach that evaluates systems by examining only their input-output behavior, treating models as black boxes. Ribeiro et al. (Ribeiro et al., 2020) showcases how the CheckList framework applies behavioral testing principles with three NLP tasks: sentiment analysis, which identifies the emotional tone (positive, negative, or neutral) of text; duplicate question detection (QQP), which determines whether two questions have the same meaning; and machine comprehension (MC), which assesses a model's ability to understand text and answer related questions. The CheckList framework was evaluated by applying it to Microsoft's commercial sentiment analysis model, uncovering previously missed bugs and enabling deeper testing of its capabilities. Additionally, a user study with 18 industry and academic participants demonstrated CheckList's practical value by successfully identifying significant issues in a Duplicate Question Detection (QQP) model. We will apply the checklist framework to Semantic Role Labelling (SRL).

One can use a CheckList matrix to display results of failure rates on the tests: capabilities such as Vocabulary+POS, Fairness and Temporal are represented as rows, while different test types - Minimum Functionality Tests (MFT), Invariance Tests (INV), and Directional Expectation Tests (DIR) - are organized as columns. Users aren't required to fill every cell, but rather use this matrix as a guide to ensure a wide evaluation of various language behaviors. Test cases can be created manually, by modifying existing datasets, or masked language models like RoBERTa can be employed to automatically generate numerous test examples by providing fill-ins for template sentences.

Consider applying CheckList to a sentiment analysis task (Ribeiro et al., 2020). Under the Vocabulary+POS capability with a Minimum Functionality Test (MFT), one might evaluate if the model correctly assigns neutral sentiment to sentences like "The company is Australian" or negative sentiment to clearly negative statements like "I despised that aircraft." For an Invariance Test (INV) assessing Robustness, slight modifications, such as adding typos "@JetBlue" "@JeBtlue", should not affect the model's sentiment predictions. Additionally, a Directional Expectation Test (DIR) could verify if the sentiment appropriately becomes more positive or negative when phrases like "You are extraordinary" or "You are lame" are added.

2 Background

The capabilities suggested by Ribeiro et al. (Ribeiro et al., 2020) partially capture key aspects needed to evaluate SRL systems, though some are more directly applicable than others. Temporal understanding and negation handling are especially relevant, as SRL depends on correctly interpreting sequences of events and assigning roles even when sentence meanings are altered by negation. Vocabulary and POS tagging are also important, since identifying nouns, for instance, is useful towards accurate semantic role assignment. Explicitly including SRL as a capability isn't appropriate since SRL defines the task itself rather than being a subcapability to test. Some other capabilities, such as Taxonomy, Robustness, NER, Coreference, and Logic, may be less important for evaluating core

SRL performance, as SRL

operates at the sentence level, relying on immediate syntactic and semantic structure. However, aspects like NER and Taxonomy could still affect fairness: for instance, if an SRL system consistently fails to recognize entities or place names from certain countries or cultures, it could produce biased role labeling.

The following capabilities are proposed to further evaluate our SRL model's robustness in handling real world inputs. Distance from subject assesses the model's ability to maintain role accuracy in long sentences, such as with parenthetical clauses, which are common in natural language. Spatial vs temporal understanding ensures the model can distinguish between modifiers like "on Tuesday" and "on top of a mountain," which carry different semantic implications. Predicate sense disambiguation evaluates how well the model handles verbs with multiple meanings, such as "I was delighted" versus "he delights in," where semantic roles differ despite the same verb. Dative alteration checks whether the model assigns consistent roles across syntactic variations with the same meaning, such as "I gave Jack a toy" and "I gave a toy to Jack." Finally, head extraction examines the model's ability to process complex noun phrases with multiple modifiers, which can complicate role identification. These capabilities are not universally required for all SRL evaluations, but we choose to test them as they target areas that are common and where, intuitively, the models may struggle in accuracy. We will assess performance per test using a failure rate, which measures the proportion of test examples where the model's output did not meet the expected criteria. *See Table 1 for proposed checks, explanations, and examples.*

In order to guide the creation of our capability tests, PropBank (02) is particularly useful because it provides detailed annotations of predicates along with their argument structures and roles. For predicate sense disambiguation, PropBank can help identify verbs with multiple senses and corresponding role sets, allowing us to construct test cases that require the model to correctly distinguish between these senses based on context. Similarly, for dative alternation, PropBank can guide us in selecting predicates that support both syntactic forms (e.g., "give Jack a toy" vs. "give a toy to Jack") while maintaining consistent semantic roles.

3 Creating a Challenge Dataset

To create our challenge dataset, we first need to define how our standalone functions will assess model performance. For MFT tests, we require a tokenized sentence, the index of the predicate, the target semantic role label , and the index of the target word. For INV tests, we need two tokenized sentences that share the same predicate and target SRL, with only the target word positions differing. For DIR tests, we also use two tokenized sentences with the same predicate, but here both the SRLs and their positions can differ. To assist in generating the raw data, we will use ChatGPT-o3-mini-high to produce CSV files containing untokenized sentences, predicate word indexes, and target word indexes. We will then manually verify each sentence and index, and use a script to tokenize the sentences, adjust the indexes accordingly, and add any additional information needed for evaluation. Please refer to appendix for complete ChatGPT-o3-mini-high prompts and dates.

After manually inspecting the dataset, 3 sentences were removed and 7 indexing errors were identified and corrected. Following this, we can create a script that uses spaCy to tokenize each sentence, map the updated indexes of the predicate and target words, and organize the data into a JSON file structured by capability → test → test type, with each entry containing the required arguments (e.g., tokenized sentence, predicate position, etc.).

For the special case of DIR (predicate disambiguation), we require two tokenized sentences with different predicate positions. One sentence should include an additional semantic role label (SRL), while the other should not. This dataset must be created manually by consulting PropBank (02) to identify predicates with multiple rolesets that take different semantic roles. For each such predicate, we construct two sentences: one using the roleset that includes the additional argument, and another using a roleset that does not, but where an added clause creates ambiguity to challenge the model's ability to disambiguate based on context. For example:

- *(Assemble as "to bring together" – does not take ARG4 goal state)* "He assembled enemies into a united group." **No ARG4**
- *(Assemble as "to build" – takes ARG4 goal state)* "John's assembly of the furniture from a kit to a home." **ARG4 present**

	Test Type + Description	Example Test Cases & Expected Behavior
Distance.	**MFT:** Test assignment of ARG0 when a parenthetical phrase separates the subject and predicate.	"Jack, despite having an extremely busy schedule and numerous responsibilities, fixed the bike." **Expected:** Predicate "fixed" → "Jack" = ARG0
	MFT: Test consistent ARG0 assignment across multiple predicates.	"John kicked the ball and scored a goal." **Expected**: Predicate "scored" → "John" = ARG0
SpaceTemp.	**DIR:** Test that switching a modifier from temporal to spatial yields different adjunct roles.	(a) "John met Jack on Friday." **Expected**: Predicate "met" → "Friday" = ARGM-TMP (b) "John met Jack under a bridge." **Expected**: Predicate "met" → "bridge" = ARGM-LOC
PredicateDis.	**DIR:** Test that the presence or absence of an additional argument distinguishes verb senses.	(a) "Maria entertained moving to Italy with Mark." **Expected**: Predicate "entertained" → "Mark" = O (b) "Maria entertained her guests with food." **Expected**: Predicate "entertained" → "food" = ARGM-MNR
Dative.	**INV:** Verify that active and prepositional constructions consistently label the instrument/benefactive/attribute as ARG2.	(a) "I gave John a book." (b) "I gave a book to John." **Expected**: Predicate "gave" → "John" = ARG2
Negation.	**MFT:** Test that explicit negation is captured.	"Jack did not clean the house." **Expected**: Predicate "clean" → 'not" = ARGM-NEG
HeadExtract.	**MFT:** Test that the model correctly identifies the head noun from a noun phrase with a single (participial) adjective.	"The painted door creaked." **Expected**: Predicate "creaked" → "door" = ARG0
	MFT: Test that the model correctly identifies the head noun from a noun phrase with multiple (participial) adjectives.	"The old, abandoned building fell." **Expected**: Predicate "fell" → "building" = ARG0

Table 1: Challenge Tests for SRL Capabilities

We create 6 such sentence pairs and store them in our JSON file using the following structure: tokenized1, tokenized2, predicate_index1, predicate_index2, and SRL. We then check whether the SRL is absent in tokenized1 and present in tokenized2, without verifying the exact position of the argument.

4 Model Descriptions

4.1 Logistic Regression

Logistic regression is a classification model that maps input features to predicted labels using a weighted sum and a sigmoid function (04, 11:32:40+00:00). For each token in a sentence, we extract a set of features and combine them into a single string, which is then one-hot encoded, transforming each unique feature combination into a binary vector. The target labels, semantic roles like ARG0, ARG1, or O, are label encoded into integers. During training, the model learns weights for each feature to estimate the probability that a token belongs to each role. It does this by applying a sigmoid function to the weighted input, which compresses the output to a value between 0 and 1. For this multi-class classification problem, the model selects the label with the highest predicted probability. At prediction time, the model processes the one-hot encoded feature vector, applies the sigmoid function, and outputs a predicted label, which is then decoded back to the original semantic role.

The features used aim to capture both syntactic and semantic information relevant to SRL with three features. The complex feature encodes the dependency path from each token to the predicate using directional arrows and includes the predicate's lemma. This helps the model understand the structural relationship between a token and the predicate. The lemma of each token is included to capture some word semantics, allowing the model to recognize semantically similar tokens such as "run," "runs," and "running" as playing similar roles, even if their surface forms differ. Lastly, the part-of-speech (POS) tag gives information about the grammatical category of the token, which supports the task since, for example, nouns often act as arguments while adjectives typically do not. All of these features are combined, one-hot encoded,

and fed into the logistic regression model to learn patterns useful for accurate semantic role labeling.

On the held-out test set (03), the model showed strong precision but lower recall across most semantic roles. For core roles like ARG0, ARG1, and ARG2, precision ranged from 0.86 to 0.91, while recall was more limited at around 0.50–0.52, indicating the model was good at predicting these roles when it made a decision, but often missed them. ARG3 had perfect precision (1.0) but very low recall (0.2), suggesting it was rarely predicted but almost always correct when it was. ARG4 showed moderately high precision (0.79) with a recall of 0.47. Among the adjunct roles, labels like ARGM-TMP, ARGM-MNR, ARGM-LVR, ARGM-EXT, ARGM-ADJ, ARGM-LOC and ARGM-ADV also displayed high precision but lower recall. For ARGM-MOD and ARGM-NEG, the model achieved both high precision and moderately higher recall, around 0.5–0.6, reflecting more balanced performance. Overall, the model tended to be conservative in its predictions, favoring precision over recall; this is reflected in the overall macro average scores, with a precision of 0.49, recall of 0.17, and F1 score of 0.23

From these results, I expect the model to perform strongly on the negation task (ARGM-NEG) and tasks involving identification of ARG0, such as head extraction and distance from the predicate, as it demonstrated consistently high precision for these roles in the test set. I anticipate moderate performance on the spatial-temporal task, given its test set results on ARGM-LOC and ARGM-TMP, which showed high precision but lower recall. Similarly, performance on the dative alternation task is expected to be moderate, as it primarily concerns ARG4, for which the model achieved reasonable precision but struggled with recall. For predicate sense disambiguation, I do not expect the model to perform well. In this task, each test case involves a different sentence using the same predicate but with different different rolesets—one may involve ARG3, while another does not. The challenge lies in correctly identifying which roles are relevant based on the predicate's sense in context. Since our model relies mostly on syntactic features, I believe it lacks the deeper semantic understanding needed to differentiate between predicate meanings. While the lemma feature provides some surface-level semantic information, it is not sufficient for identifying subtle shifts in meaning that affect role presence, making this a particularly difficult task for the logistic regression setup.

4.2 DistilBert

We utilize "distilbert-base-uncased," (06) a smaller and more efficient version of BERT (Bidirectional Encoder Representations from Transformers). BERT's architecture consists of multiple layers of transformers that process input text to generate contextual embeddings, capturing the meaning of words based on their context within a sentence (Kumari, 2023). DistilBERT retains this transformer architecture but reduces the number of layers and uses knowledge distillation to create a faster model that maintains much of BERT's performance. To adapt DistilBERT for semantic role labeling (SRL), we replace its original pretraining head with a token classification head (08). This new head is a linear layer that maps the contextual embeddings of each token to a probability distribution over possible semantic roles, enabling the model to assign appropriate roles to each token in a sentence.

For predicate marking, we adopt a modified approach inspired by Shi and Lin (2019) (Shi and Lin, 2019). Instead of appending the predicate to the end of the sentence, we apply the following steps:

1. Split the original sentence at the predicate position.

2. Insert a special `[PRED]` token immediately after the predicate token.

3. Tokenize the resulting sequence and process it through the model.

This method explicitly marks the predicate whose arguments we aim to identify, reducing risk of ambiguous predicate assignment. When handling multiple predicates in a sentence, we create separate instances of the sentence, each with a different predicate marked with the [PRED] token.

On the held-out test set (03), the fine-tuned transformer model demonstrated strong and consistent performance on core semantic roles. ARG0, ARG1, and ARG2 achieved high precision and recall, with F1-scores between 0.77 and 0.87, reflecting the model's ability to reliably identify the most frequent argument roles. As with the previous model, ARG3 remained challenging: despite a perfect precision score of 1.0, recall was very low (0.03), indicating the model rarely predicted this

role. ARG4 performed better than in the logistic regression model, with an F1-score of 0.58, showing improvement but still suggesting limited generalization to less common argument types. Among adjunct roles, performance varied more widely. Common roles like ARGM-TMP, ARGM-MOD, and ARGM-NEG were predicted with high precision and recall (F1-scores above or equal 0.83), suggesting the model effectively learned to recognize temporal expressions and common modifiers. Roles such as ARGM-ADV, ARGM-LOC, and ARGM-LVB also achieved reasonably balanced scores. Overall, the model displayed reasonably balanced precision and recall across classes, as reflected in its macro average scores: 0.48 for precision, 0.41 for recall, and 0.42 for F1, highlighting its ability to generalize beyond just the most frequent labels.

From these results, similar to the Logistic Regression model, I expect the transformer-based model to perform strongly on the negation task (ARGM-NEG) and tasks involving the identification of ARG0, such as head extraction and distance from the predicate, given its consistently high precision and recall for these roles in the test set. I also anticipate strong performance on the spatial-temporal task, supported by solid F1-scores for both ARGM-LOC and ARGM-TMP. Performance on the dative alternation task is expected to be moderate to good, as this involves ARG4, for which the model showed a relatively balanced precision (0.61) and recall (0.56), and an improved F1-score compared to the Logistic Regression model. For predicate sense disambiguation, I expect the model to show some accuracy with limitations. Unlike the logistic model, the transformer can leverage contextual embeddings to distinguish between different senses of the same predicate when those differences are reflected in sentence structure or context. However, the model is still limited by the absence of explicit predicate sense annotations, which may hinder its ability to consistently select the correct roleset across subtle variations in meaning.

5 Results

We present the following results retrieved from test_challenge.ipynb, displaying the capability, test (in same order as Table 1) and failure rate.

Capability	Test	Failure Rate
Distance	MFT1	79.07%
	MFT2	100%
SpaceTemp.	DIR	100%
DativeAlter.	INV	66.67%
Negation	MFT	68%
HeadExtract.	MFT1	90%
	MFT2	96%
PredicateDis.	DIR	100%

Table 2: Failure rates by capability and test type (Logistic Regression)

Capability	Test	Failure Rate
Distance	MFT1	4.65%
	MFT2	6%
SpaceTemp.	DIR	13.56%
DativeAlter.	INV	21.05%
Negation	MFT	0%
HeadExtract.	MFT1	58%
	MFT2	48%
PredicateDis.	DIR	100%

Table 3: Failure rates by capability and test type (DistilBERT)

6 Discussion

The Logistic Regression model performed surprisingly poorly on Distance and Head Extraction, both of which involve identifying ARG0 a role the model handled well in the held-out test set. This discrepancy suggests that while the model can recognize ARG0 in straightforward syntactic structures, it fails when ARG0 is separated from the predicate (e.g., by a parenthetical). Similarly, its 100% failure rate on SpaceTemp was unexpected, given moderate test set performance on ARGM-LOC and ARGM-TMP. This capability required distinguishing between spatial and temporal adjuncts in otherwise similar sentences, and the model may not have learned the semantic cues needed to separate them when they appear in parallel positions. On Negation, it failed 68% of the time, despite an F1 score of 0.69 for ARGM-NEG in evaluation, suggesting the model may only handle the most simple forms of negation and lacks robustness in varied expressions.

The best relative performance was on Dative Alternation, where the model failed 66.67% of the time. This result is still weak, but aligns with our prediction that performance would be moderate, since ARG4 showed reasonable precision but poor recall in training. The model may have benefited from consistent syntactic patterns across both active and prepositional forms. On Predicate Disambiguation, the model failed all cases (100%), which

confirms expectations, its syntactic feature set is insufficient for distinguishing rolesets that depend on deeper semantic interpretation. However, this result is based on only six manually created sentence pairs, limiting the reliability of this conclusion. A larger, more varied dataset would be needed to assess whether the model's failure is consistent or specific to the test cases used.

The DistilBERT model performed well on several capabilities, especially where we expected stronger generalization from contextual understanding. Most notably, it achieved 0% failure on Negation, successfully identifying ARGM-NEG in every test case. This aligns with its high precision and recall in the test set and demonstrates the model's ability to handle diverse expressions of negation, an area where the Logistic Regression model struggled. Performance on Distance tasks was also strong, with failure rates of just 4.65% and 6%, confirming its robust handling of ARG0 even when it is syntactically distant from the predicate. This suggests the transformer is effectively leveraging contextual cues beyond surface-level patterns. On SpaceTemp, the model had a modest 13.56% failure rate, which is within expectations and indicates a reasonable ability to distinguish between ARGM-TMP and ARGM-LOC despite their structural similarity. Compared to Logistic Regression's 100% failure, this highlights the advantage of contextual embeddings for resolving adjunct roles. Dative Alternation also showed improvement, with a 21.05% failure rate, suggesting that DistilBERT is better able to generalize across alternations involving ARG4.

However, performance on Head Extraction was lower than anticipated, with failure rates of 58% and 48%. While the model performed well on ARG0 in general, these results suggest difficulty with identifying heads within complex noun phrases, especially when adjectives or participial modifiers are involved. This indicates a potential blind spot in how the model processes internal phrase structure, where contextual cues alone may not fully compensate for structural complexity. Predicate Disambiguation proved to be the most challenging task, still with a 100% failure rate. Although the model has access to contextual information, it lacks the explicit predicate sense annotations necessary to reliably distinguish rolesets. The six manually constructed sentence pairs offer only a limited sample, but the consistent failure indicates that even transformer-based models may struggle with this deeper semantic task when subtle shifts in meaning determine role presence.

7 Future Work

To build a system an SRL system for doctor-patient conversations, we start by training on a large general SRL dataset (Universal PropBank) to learn broad language patterns. Then we fine-tune the model using the small amount of labeled data we have from real GP appointments. To make better use of the larger unlabeled corpus, we apply methods like self-training (Steen, 2020), where the model labels new in-domain examples and we keep the most confident predictions.

To evaluate how well the system performs in this specific setting, we design a test set using the CheckList framework and define capabilities. For example, we might test if the model consistently identifies symptoms or medications across different sentence forms (MFT), handles rewordings of the same complaint (INV), or adjusts its role labels when a condition improves or worsens (DIR). This gives us a clearer view of how well the model understands real GP dialogue.

We create these test cases using both real and generated examples, focusing on challenges specific to clinical conversations such as handling shifts in who is being talked about, like switching from the patient's symptoms to the doctor's instructions. We then track failure rates for each capability, helping us see exactly where the model struggles.

8 Conclusion

Applying the CheckList framework revealed critical insights into SRL model performance that standard held-out test sets failed to show. While both Logistic Regression and DistilBERT scored well on test data, this behavioral testing uncovered weaknesses that affect real-world reliability. For instance, the Logistic Regression model showed high ARG0 precision in the test set, yet failed on Distance and Head Extraction tasks—highlighting its inability to handle syntactic variation. It also struggled with semantic distinctions, failing 100% of SpaceTemp and Predicate Disambiguation tests, despite reasonable test performance on ARGM-LOC, ARGM-TMP, and ARGM-NEG. These limitations were not evident from aggregate metrics. DistilBERT, by contrast, showed robustness in handling

varied negation and long-distance dependencies, but unexpectedly underperformed on Head Extraction and failed entirely on Predicate Disambiguation—revealing gaps even in contextual models. CheckList goes beyond surface accuracy to assess actual linguistic competence. It enables evaluate if our models are equipped for complex, real-world language.

References

DistilBERT. https://huggingface.co/docs/transformers/en/model_doc/distilbert.

Notebooks/examples/token_classification.ipynb at main · huggingface/notebooks. https://github.com/huggingface/notebooks/blob/main/examples/token_classification.ipynb.

The Proposition Bank (PropBank). https://propbank.github.io/.

Universal PropBank. https://github.com/UniversalPropositions.

11:32:40+00:00. Logistic Regression in Machine Learning. https://www.geeksforgeeks.org/understanding-logistic-regression/.

Priyanka Kumari. 2023. BERT and DistilBERT Models for NLP. https://medium.com/@kumari01priyanka/bert-and-distilbert-model-for-nlp-7352eb16915e.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Peng Shi and Jimmy Lin. 2019. Simple BERT Models for Relation Extraction and Semantic Role Labeling.

Doug Steen. 2020. A Gentle Introduction to Self-Training and Semi-Supervised Learning. https://towardsdatascience.com/a-gentle-introduction-to-self-training-and-semi-supervised-learning-ceee73178b38/.

Appendix: Dataset Generation Prompts

Distance

MFT: Test assignment of ARG0 when a parenthetical phrase separates the subject and predicate. chatgpt-o3-mini-high 23/03/2025

Please generate 100 distinct sentence examples that begin with a subject (ARG0) and include a simple direct object (ARG1), with a parenthetical phrase inserted between the subject and the predicate. Use varied length and complexity for the parenthetical phrase, and ensure each sentence is semantically correct.

Output the results in a CSV file with three columns:

- the full sentence,
- the word index (based on .split(" ") and excluding punctuation) of the significant word in the subject (ARG0),
- the word index (again excluding punctuation) of the predicate verb, which comes immediately after the parenthetical phrase.

Note: If the subject consists of two words (e.g., "My neighbor"), use only the significant word (e.g., "neighbor") for the ARG0 index. Do not include commas or other punctuation when determining word indexes.

Example:
Sentence: "Jack, despite having an extremely busy schedule and numerous responsibilities, fixed the bike."
ARG0 index: 0 (for "Jack")
Predicate index: 10 (for "fixed")

MFT: Test consistent ARG0 assignment across multiple predicates. chatgpt-o3-mini-high 23/03/2025

Please generate 100 distinct sentence examples in which a single subject (ARG0) performs two or three different actions using two or three different predicate verbs (e.g., "kicked" and "scored"), each with a direct object (ARG1). Ensure the subject clearly governs all verbs, and the sentence is semantically correct.

Output the results in a CSV file with three columns:

- the full sentence,
- the word index (based on .split(" ") and excluding punctuation) of the significant word in the subject (ARG0),
- the word index of the last predicate verb (excluding punctuation).

Note:

- If the subject consists of two words (e.g., "My sister"), only count the **significant word** (e.g., "sister") as the ARG0 for indexing.
- Do not include commas or other punctuation when determining word indexes.
- The last verb should occur after a conjunction like “and” or “then” to show coordination.

Example:
Sentence: "John kicked the ball and scored a goal."
ARG0 index: 0 (for "John")
Second predicate index: 5 (for "scored")

SpaceTemp

DIR: Test that switching a modifier from temporal to spatial yields different adjunct roles. chatgpt-o3-mini-high 23/03/2025

Please generate 50 distinct pairs of sentence examples that begin with a subject (ARG0) and include a predicate verb, followed by both a temporal modifier (ARGM-TMP) and a spatial modifier (ARGM-LOC). Ensure each pair of sentences is semantically correct and clearly demonstrates the difference between temporal and spatial adjunct roles.

Output the results in a CSV file with five columns:

- the full sentence with the temporal modifier (SentenceTemp),
- the full sentence with the spatial modifier (SentenceLoc),
- the word index (based on .split(" ") and excluding punctuation) of the predicate,
- the word index (again excluding punctuation) of the significant word in the temporal modifier (ARGM-TMP),
- the word index (again excluding punctuation) of the significant word in the spatial modifier (ARGM-LOC).

Example:
SentenceTemp: "John met Jack on Friday."
SentenceLoc: "John met Jack under a bridge."
Predicate index: 1 (for "met")
ARGM-TMP index: 4 (for "Friday")
ARGM-LOC index: 5 (for "bridge")

DativeAlter

INV: Verify that active and prepositional constructions consistently label the recipient (ARG2). chatgpt-o3-mini-high 23/03/2025

Please generate 50 distinct sentence examples that have a recipient, instrument or benefactive attribute (ARG2). Ensure each sentence is semantically correct and clearly demonstrates the difference between active and prepositional constructions; add variation to the subject, objects, predicates and syntactical structure of the sentence.

Output the results in a CSV file with five columns:

- the full sentence in active construction (SentenceActive),
- the full sentence in prepositional construction (SentencePrepositional),
- the word index (based on .split(" ") and again excluding punctuation) of the predicate verb,
- the word index (again excluding punctuation) of the ARG2 for SentenceActive,
- the word index (again excluding punctuation) of the ARG2 for SentencePrepositional.

Example:
SentenceActive: "I gave John a book."
SentencePrepositional: "I gave a book to John."
Predicate index: 1 (for "gave")
Active ARG2 index: 2 (for "John")
Prepositional ARG2 index: 5 (for "John")

Negation

MFT: Test that the model correctly identifies explicit negation (ARGM-NEG). chatgpt-o3-mini-high 23/03/2025

Please generate 50 distinct sentence examples that include an explicit negation (e.g., "did not", "can not", "not"). Add sufficient variation in the formation of the sentences.

Output the results in a CSV file with three columns:

- the full sentence,
- the word index (based on .split(" ") and excluding punctuation) of the predicate,
- the word index (again excluding punctuation) of the ARGM-NEG.

Example:
Sentence: "Jack did not clean the house."
Predicate index: 3 (for "clean")
ARGM-NEG index: 2 (for "not")

HeadExtract

MFT: Test with a noun phrase containing a single (participial) adjective. chatgpt-o3-mini-high 23/03/2025

Please generate 50 distinct sentence examples that include a noun phrase with a single adjective or participial adjective and a predicate verb. Ensure each sentence is semantically correct; add variation to the subject, objects, predicates and syntactical structure of the sentence.

Output the results in a CSV file with three columns:

- the full sentence,
- the word index (based on .split(" ") and excluding punctuation) of the predicate,
- the word index (again excluding punctuation) of the ARG0.

Example:
Sentence: "The painted door creaked."
Predicate index: 3 (for "creaked")
ARG0 index: 2 (for "door")

MFT: Test with a noun phrase containing multiple (participial) adjectives. chatgpt-o3-mini-high 23/03/2025

Please generate 50 distinct sentence examples that include a noun phrase with multiple adjectives or participial adjectives and a predicate verb. Ensure each sentence is semantically correct; add variation to the subject, objects, predicates and syntactical structure of the sentence.

Output the results in a CSV file with three columns:

- the full sentence,
- the word index (based on .split(" ") and excluding punctuation) of the predicate,
- the word index (again excluding punctuation) of the ARG0.

Example:
Sentence: "The old, abandoned building fell."
Predicate index: 4 (for "fell")
ARG0 index: 3 (for "building")