Leveraging pre-trained language models for linguistic analysis: A case of argument structure constructions

Hakyung Sung and **Kristopher Kyle**
Learner Corpus Research and Applied Data Science Lab
Department of Linguistics
University of Oregon
hsung,kkyle2@uoregon.edu

Abstract

This study evaluates the effectiveness of pre-trained language models in identifying argument structure constructions, important for modeling both first and second language learning. We examine three methodologies: (1) supervised training with RoBERTa using a gold-standard ASC treebank, including by-tag accuracy evaluation for sentences from both native and non-native English speakers, (2) prompt-guided annotation with GPT-4, and (3) generating training data through prompts with GPT-4, followed by RoBERTa training. Our findings indicate that RoBERTa trained on gold-standard data shows the best performance. While data generated through GPT-4 enhances training, it does not exceed the benchmarks set by gold-standard data.

1 Introduction

Argument structure constructions (ASCs) are lexicogrammatical patterns at the clausal level. They consist of an argument structure and a main verb, with each argument contributing to the clause's meaning (Goldberg, 1995). The characteristics of ASC use, such as frequency and/or the strength of association between a verb and its argument structure, have been actively explored in previous studies on first language (L1) and second language (L2) learning and assessment (Tomasello and Brooks, 1998; Ellis, 2002; Ninio, 1999; Kyle and Crossley, 2017).

To effectively model human language learning/development using ASC features, ASCs must be reliably identified in target texts. Recent studies have shifted from manual (e.g., Ellis and Ferreira-Junior, 2009) to automatic ASC analyses (e.g., Kyle, 2016; Kyle and Sung, 2023; Hwang and Kim, 2023). However, building automated ASC annotation systems has proven challenging due to syntactic ambiguity. For example, while syntactic analyses would represent the clauses (1) *she ran [to the mountains]* and (2) *she ran [in the mountains]* identically based on the form (i.e., subject-verb-prepositional phrase structures), they imply different meanings and represent distinct ASC types. In case (1), the prepositional phrase [*to the mountains*] is an argument that completes the meaning by specifying the goal of the movement. In contrast, in case (2), the phrase [*in the mountains*] modifies the location of the event, as illustrated in Figure 1. One potential reason for this mismatch is that human language often employs a pre-built form-meaning schema at the clausal level (Fillmore, 1968; Goldberg, 1995), which can be challenging to capture from a bottom-up perspective. A top-down approach, directly assigning ASC types based on their clausal contexts, is therefore likely more effective than a bottom-up approach.

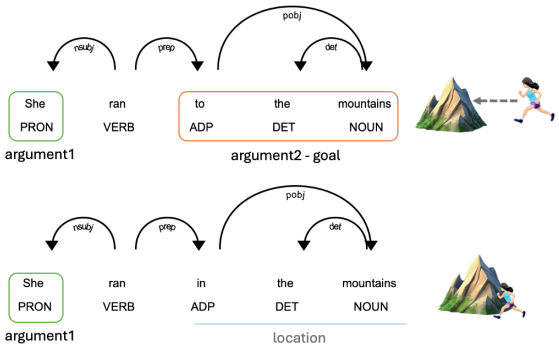

Figure 1: Distinguishing semantic roles in similar dependency structures of two different types of ASCs, visualized by *DisplaCy* (Honnibal et al., 2020)

Recent advancements in pre-trained language models (PLMs) may offer a promising solution to these challenges, given their effectiveness in storing sentence-level contextual knowledge, as well as part-of-speech and syntactic knowledge within their word embeddings (Miaschi and Dell'Orletta, 2020; Hewitt and Manning, 2019). The follow-up empirical question is whether these models can reliably capture specific types of ASCs, both with and

without top-down annotations provided by trained human annotators focusing on the linguistic characteristics of clausal forms. To address this, the current study explores the use of PLMs for identifying ASCs, evaluating three methodologies: (1) supervised training with a encoder model (RoBERTa) using a gold-standard ASC treebank, (2) prompt-guided annotation of unlabeled data with a decoder mode (GPT-4), and (3) prompt-guided generation of training data with GPT4, followed by training with RoBERTa.

2 Backgrounds

2.1 Language learning and ASC use

Usage-based constructionist theories propose that language development occurs as learners form form-meaning pairings (i.e., constructions) through statistical induction from diverse linguistic inputs. In modeling language learning/development, a key aspect of this approach involves ASCs, which are clausal level constructions that convey the core concepts of a sentence. They are also instrumental in communications as they encapsulate conceptual archetypes, such as motion or causative events (Bencini and Goldberg, 2000; Goldberg, 1995, 2003; O'Connor and Kay, 2003; Rappaport Hovav and Levin, 1998).

Building on theories that emphasize the significance of ASCs, empirical studies in language learning (e.g., Ellis, 2002; Ninio, 1999; Tomasello, 2005; Gries and Wulff, 2005) have indicated that the frequency of ASCs (and of verbs), along with the strength of their associations, are key factors in shaping their developmental trajectory. To be specific, language learners make form-meaning mappings between frequent linguistic forms (e.g., ***give***-*me-the-toy*) and their corresponding meanings (Ninio, 1999) early in their learning process. As learners encounter more related but varied inputs (e.g., ***hand***-*me-the-toy*, ***bring***-*me-the-toy*), they develop schematic representations of these forms like ***VERB***-*me-the-toy*, or more abstractly, ***VERB***-*Recipient-Object*)[1]. In short, as they develop, learners adopt a broader range of less frequent ASCs, utilize a wider range of verbs within specific ASCs, and form stronger associations between verbs and ASCs, thus reducing their use of atypical verbs in these constructions.

The use of ASCs has proven to be a useful indicator of language proficiency, applicable to NLP applications such as automatic scoring and modeling human language development. Kyle and Crossley (2017), for example, found that more proficient L2 writers tended to use less frequent but more strongly associated verb-ASC combinations. Additionally, they found that ASC-based indices were better predictors of holistic writing scores than classic indices of syntactic complexity (e.g., mean length of T-unit, mean length of clause), which focused on the structural elements of sentences without accounting for the functional relationships conveyed by ASCs. Relatedly, scholars have also found that the use of particular ASC types indicate L2 proficiency. For example, Hwang and Kim (2023) found that more proficient L2 writers tended to use a wider range of ASC types overall, and also tended to use a higher proportion of passive and caused-motion ASCs.

2.2 Identification of ASCs

To accurately and reliably identify ASCs, initial studies relied on time-intensive manual analyses (e.g., Ellis and Ferreira-Junior, 2009). However, recognizing the need for efficiency, researchers have increasingly been investigating the feasibility of automated ASC analysis for some time now, as illustrated below.

2.2.1 Use of dependency representations

The advent and popularization of syntactic dependency representation in treebanks and parsers (de Marneffe and Manning, 2008; Chen and Manning, 2014) provided a helpful starting point for automated ASC analysis. For example, O'Donnell and Ellis (2010) used a dependency parsed version of the BNC (Andersen et al., 2008) to explore the feasibility of extracting ASCs using dependency tags. While this approach allowed for some target constructions to be accurately extracted (e.g., VERB-preposition-noun construction: [*talked about it*], Römer et al., 2014) overall accuracy was insufficient for broader use.

Over time, the introduction of NLP systems utilizing neural networks (e.g., Chen and Manning, 2014) substantially increased dependency parsing accuracy, sparking renewed efforts in automated ASC annotation (e.g., Hwang and Kim, 2023; Kyle,

[1]Research has shown that learners tend to initially overgeneralize schematic slots (Ambridge et al., 2013; Goldberg et al., 2004; Ninio, 1999). For example, after learning how to use a basic transitive ASC form (e.g., *she opened the door*), a learner might mistakenly extend this construction to intransitive verbs, resulting in ungrammatical sentences (e.g., *she sits the chair*). However, they gradually fine-tune their linguistic system through additional input and use.

2016; Kyle and Crossley, 2017). In summary, with these more accurate parsers, researchers attempted to extract ASCs based on syntactic frames (built from dependency representations) and employed mapping systems to categorize them. However, a key issue with using only syntactic frames to identify ASCs is that current representations lack the semantic information needed to disambiguate certain ASC types (e.g., between intransitive simple and intransitive motion constructions, as illustrated in Figure 1; also see Kyle and Sung, 2023). To improve accuracy, an alternative approach that considers the semantics of the clause is necessary.

2.2.2 Use of semantic role labels

Another approach involves leveraging the semantic information in ASCs, as they contain "argument roles" that often correspond to traditional semantic roles (e.g., agent, patient, theme, goal). For scalable and automatic extraction of ASCs, databases annotated with semantic role labels, such as PropBank (Palmer et al., 2005) or the Universal Propositions (UP) treebank (Akbik et al., 2015), or automated semantic role labeling systems(Gardner et al., 2018; Shi and Lin, 2019), may prove useful. This approach mirrors previous efforts where researchers extracted ASCs based on syntactic information from dependency treebanks and parsers.

However, there are two major obstacles with solely relying on semantic role labelling systems at present. First, the accuracy of current automated semantic role labeling systems is still not sufficient for this task[2]. Second, it is sometimes not straightforward to map the output of semantic role labeling systems to ASCs. Typically, these systems use abstract semantic role labels (e.g., ARG0, ARG1) that pose challenges in directly mapping to theoretical ASC categorizations for some complex ASCs[3].

To address this issue, one potential solution involves automatically extracting semantic roles from a clause, grouping them into semantic frames, and mapping each frame to corresponding ASC types using linguistic knowledge. Subsequently, these semi-automatically mapped ASCs can be trained using a sequential learning model. For example, Kyle and Sung (2023) utilized a combination of UP treebank (Akbik et al., 2015), VerbNet (Schuler, 2005), and FrameNet (Fillmore et al., 2003) to extract semantic roles and corresponding sentences from a subset of the English Web Treebank (Silveira et al., 2014). The extracted semantic roles were grouped into semantic frames, and researchers assigned ASCs to each frame, creating a silver-annotated ASC treebank (ASC Treebank v1). They then trained a transformer model using RoBERTa embeddings with the semi-automatically annotated ASC labels and compared its performance against three probabilistic models: one based on verb lemmas, another on syntactic frames using dependency parsing, and a third combining both verb lemmas and syntactic frames. The results indicated that the transformer model trained on silver-annotated sentences based on the semantic role labels achieved the highest classification accuracy (F1 = .918), outperforming the other models. Despite this success, there is room to improve the model's accuracy (particularly at the ground truth level) by leveraging gold-standard annotations (Kyle and Sung, 2023; Sung and Kyle, 2024).

2.2.3 Use of the gold standard ASC treebanks

As discussed, researchers have employed scalable databases and systems to extract ASCs. While these approaches have demonstrated effectiveness, they remain limited in their ability to differentiate ambiguous cases that cannot be fully captured by automatically extracted syntactic or semantic frames. A potential alternative is going back to construct a treebank from scratch, which echoes earlier efforts to manually identify ASCs (e.g., Ellis and Ferreira-Junior, 2009). However, the current goal is to create (1) a training set for supervised learning, specifically designed for sequential named entity recognition (NER) tasks[4], (2) input examples for few-shot learning in unsupervised tasks, and (3) a test set to evaluate the model's accuracy.

Recently, Sung and Kyle (2024) released a gold-standard annotated treebank of ASCs (ASC treebank v2), which includes sentences from the English Web Treebank (EWT), as well as sentences written by L2 users from ESL-WR (Berzak et al., 2016), and spoken by L2 users from ESL-SP (Kyle et al., 2022) (10,204 sentences; 22,069 ASC to-

[2]To our best understanding, the publicly-available semantic role labeling achieved an F1 of 0.86 on argument tagging, and 0.95 on predicate tagging (Gardner et al., 2018; Shi and Lin, 2019). Note that these scores are for large-grained argument tags, which do not offer the precision required for ASC identification.

[3]Particularly, ARG2 and ARG3 cover a number of semantic categories. According to Jurafsky and Martin (Chapter 24.4), ARG2 includes benefactive, instrumental, attributive, or end-state roles, while ARG3 encompasses start-point, benefactive, instrumental, or attributive roles.

[4]Methodologically, Kyle and Sung (2023) applied this approach to the silver-annotated ASC treebank.

kens). This treebank can be leveraged for more robust training and precise evaluation of developed models aimed at identifying ASCs.

3 Related work

3.1 Automated linguistic annotation with encoder models

Recent advancements have underscored the potential of PLMs in automated linguistic annotation, as encoder models (e.g., BERT [Devlin et al., 2018]; RoBERTa [Liu et al., 2019]) have demonstrated impressive gains in supervised learning tasks. Based on the Transformer architecture (Vaswani et al., 2017), PLMs have been extensively pre-trained on large text corpora and adeptly store morpho-syntactic and sentence-level contextual knowledge within their word embeddings (Miaschi and Dell'Orletta, 2020; Hewitt and Manning, 2019). One fundamental application, often considered first in linguistic annotation, is dependency tagging and parsing. The performance of these models, specified for English, typically achieves an F1 score above 0.90 (e.g., Honnibal et al., 2020). Beyond syntactic analysis, Shi and Lin (2019) demonstrated that a BERT-LSTM based model could attain F1 scores of 0.90 on in-domain test sets and 0.84 on out-domain test sets in semantic role labeling. This was accomplished through argument identification and classification, without the need for auxiliary syntactic features like part-of-speech tags or dependency trees.

While dependency parsing and semantic role labeling have focused on word-level linguistic annotations, a RoBERTa-based model has shown promising results with discourse-level linguistic annotation. For example, recently Eguchi and Kyle (2023) applied a RoBERTa-based ensemble model to identify and categorize rhetorical stance features in academic English writing. By employing a discourse analytic framework and manually annotating 4,688 sentences across eight rhetorical stance categories, they trained an ensemble model combining RoBERTa and LSTM. This model achieved a macro-averaged F1 score of 0.72 in span identification of stance-taking expressions, surpassing pre-adjudication human annotator reliability.

3.2 Automated linguistic annotation with decoder models

To effectively employ encoder models for fine-grained linguistic analyses, it is important to collect and precisely annotate a certain amount of training data for the linguistic features of interest. However, data annotation is often a costly process. This cost encompasses the labor involved in researchers recruiting, training, and managing human annotators, as well as the time spent by annotators in labeling raw data. In this context, recent studies have explored ways to effectively use decoder models (e.g., GPT) for data annotation with unsupervised learning (Radford et al., 2019). They have demonstrated impressive zero-shot or few-shot learning abilities, which allow them to perform tasks with minimal or no task-specific training data (Brown et al., 2020).

For example, Ding et al. (2022) investigated leveraging GPT-3 for data annotation in different NLP tasks, including an NER task. They developed three distinct GPT-3-based data annotation approaches: (1) prompt-guided unlabeled data annotation, (2) prompt-guided training data generation, and (3) dictionary-assisted training data generation. Subsequent experiments on both sequence- and token-level NLP tasks were used to evaluate their performance. The findings indicated that directly annotating unlabeled data was effective for tasks with a small labelling task, while generation-based methods proved more suitable for tasks with a larger labelling task. Similarly, Yu et al. (2023) investigates the application of GPT models to automate complex pragmatic-discourse features of apology in zero and few-shot settings. By comparing the performance of GPT-3.5, GPT-4, and human annotations in annotating apology components, the study demonstrated that GPT-4's accuracy approached that of human annotators.

On the contrary, the recent study by Ettinger et al. (2023) found limited success using GPT-3, ChatGPT, and GPT-4 models for semantic annotations (i.e., abstract meaning representation [Banarescu et al., 2013]). The experiments included zero and few-shot experiments, as well as an experiment focusing on PLMs' ability to handle metalinguistic queries (e.g., identifying primary sentence events and predicates). A comprehensive evaluation of parse acceptability demonstrated that, even with few-shot examples, the models almost never succeeded in producing completely accurate parses. The findings indicate that while these models capture some semantic elements, significant challenges persist in achieving precise semantic analyses.

4 Methodology

4.1 Datasets

In this study, we utilize two treebanks, namely the silver (v1) and gold (v2) versions of the ASC treebank. The first silver version (Kyle and Sung, 2023) includes 26,437 ASC tokens that were semi-automatically annotated[5]. The second gold version (Sung and Kyle, 2024) includes 22,069 manually annotated ASC tokens[6]. The sentences in this treebank were sampled from the English Web Treebank (ETW) (Silveira et al., 2014), L2-Written (ESL-WR) (Berzak et al., 2016), and L2-spoken (ESL-SP) (Kyle et al., 2022) treebanks, which are all part of the Universal Dependencies project (Nivre et al., 2020). Given the relatively small representation of L2-written (ESL-WR) and spoken (ESL-SP) data, training, development, and test sets were resampled with a 34/33/33 distribution. The L1 (EWT) sentences retained their original sections and were roughly distributed at 80/10/10.

Table 1 illustrates the nine ASC tags along with the most prototypical semantic roles that were mapped in two treebanks (Kyle and Sung, 2023; Sung and Kyle, 2024), accompanied by examples from the annotated dataset. Appendix A shows ASC type frequencies in each dataset.

4.2 Experiment setup

The purpose of this study is to explore how to leverage PLMs, specifically RoBERTa (an encoder model) and GPT-4 (a decoder model), for ASC annotations which could assist in modeling and measuring human language development. To achieve this goal, we designed three different approaches to utilize PLMs to evaluate and compare their performance (Figure 2).

4.2.1 Experiment 1

The objective of the first experiment is to investigate supervised learning using gold-standard data applied with RoBERTa embeddings (Liu et al., 2019). To accomplish this, we trained a transformer-based machine learning model, employing the open-access Python library, *spaCy* (version 3.7.4; Honnibal et al., 2020) for a multi-class NER task. The model leverages transformer-based RoBERTa embeddings (via the *en_core_web_trf* pipeline), integrating pre-trained embeddings and fine-tuning them for the NER task by adjusting the model's weights based on the labeled data used for training. SpaCy's method includes a transition-based parser, which is primarily used for dependency parsing but, in this case, provides syntactic context that enhances the NER model's performance.

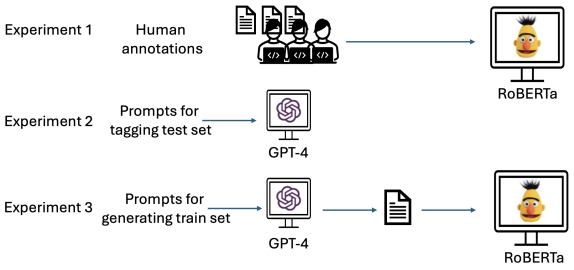

Figure 2: Experiment overview

To evaluate the performance, we constructed three comparative models: (1) a model using silver-standard data, (2) a model trained with gold L1 data, and (3) a model trained with both gold L1 and L2 data. Considering the necessity for accurate performance on L2 data to capture non-native English linguistic structures, we conducted detailed testing on each L1, L2 written, and L2 spoken data. For specifics on the hyperparameter settings, refer to Appendix B.

4.2.2 Experiment 2

The goal of the second experiment is to explore prompt-guided annotation of unlabeled data. To this end, GPT-4 was employed to generate labels for a subset of the test set from the gold-standard treebank. Due to the high processing costs and time, we streamlined the task by filtering the tag set – reducing the number of tags from nine to seven by removing the ATTR and PASSIVE tags. Moreover, we utilized a random balanced extraction method to select sentences for annotation, ultimately resulting in a total of 282 sentences.

To evaluate performance, we provided GPT-4 with three distinct prompts for label generation on the test set: (1) zero-shot, (2) 3-shot, and (3) 10-shot. In cases of few-shot learning, examples were randomly selected from the gold-standard ASC treebank (including both L1 and L2 datasets). We compared these results with baseline scores from the best model trained under a supervised learning. This comparative model, as described in Ex-

[5]This dataset (CC-BY-NA-SA 4.0) is publicly available at the ASC-Treebank GitHub repository(https://github.com/LCR-ADS-Lab/ASC-Treebank).

[6]This dataset (CC-BY-NA-SA 4.0) is publicly available at the ASC-Treebank GitHub repository and the osf repository (https://osf.io/v75qu/).

ASC (Annotated tag)	Semantic frame	Example
Attributive (ATTR)	*theme*-V-*attribute*	It_{theme} *is now* $visible_{attribute}$ *on the street*
Caused-motion (CAUS_MOT)	*agent*-V-*theme*-*destination*	I_{agent} *put* it_{theme} *[on the calendars]*$_{destination}$
Ditransitive (DITRAN)	*agent*-V-*recipient*-*theme*	I_{agent} *gave* $him_{recipient}$ *[the address]*$_{theme}$
Intransitive motion (INTRAN_MOT)	*theme*-V-*goal*	I_{theme} *won't go [out the door]*$_{goal}$
Intransitive resultative (INTRAN_RES)	*patient*-V-*result*	$Money_{patient}$ *may become* $tight_{result}$
Intransitive simple (INTRAN_S)	*agent*-V	I_{agent} *am working from the office*
Passive (PASSIVE)	*theme*-aux-$V_{passive}$	$They_{theme}$ *were* $recommended_{passive}$ *by him*
Transitive resultative (TRAN_RES)	*agent*-V-*result*-*result*	I_{agent} *don't want [my leg]*$_{result}$ $hurt_{result}$
Transitive simple (TRAN_S)	*agent*-V-*theme*	I_{agent} *should buy [a new one]*$_{theme}$

Table 1: ASCs representation

```
You are tasked with annotating English argument
structure constructions (ASCs) in sentences.
Your role is to assign one of seven different
ASC labels to verbs that form ASCs in each
sentence:
(1) transitive simple (TRAN_S),
(2) transitive resultative (TRAN_RES),
(3) caused-motion (CAUS_MOT),
(4) intransitive simple (INTRAN_S),
(5) intransitive motion (INTRAN_MOT),
(6) intransitive resultative (INTRAN_RES),
(7) ditransitive (DITRAN).
Tag only the labels. Note that a sentence may
contain multiple ASCs. Please format your
annotations in a simplified CoNLL-U format. For
each word in the sentence, list its order
number, the word itself, and its label if it is
a verb participating in an ASC; otherwise, use
an underscore (_). Example:
# text = John throws the ball.
1\tJohn\t_
2\tthrows\t[tag a label]
3\tthe\t_
4\tball\t_
5\t.\t_

Please annotate the following sentence:
```

Figure 3: Example of prompting GPT-4 to generate ASC labels in a zero-shot setting

periment 1, incorporated adaptations such as early stopping[7]. Figure 3 shows an example of a zero-shot learning.

4.2.3 Experiment 3

The objective of the third experiment is to explore the use of prompt-guided generation of training data for training RoBERTa. In this experiment, we utilized GPT-4 to create a labeled dataset, which was subsequently used to train with RoBERTa.

For data generation, GPT-4 was used to produce a balanced set of sentences with ASC tags, starting with 3-shot and 10-shot settings, as the model struggled to generate data without any initial examples. We divided the experiment into two parts: the first involved training the model solely using data generated by GPT-4; the second combined these generated sentences with a similarly balanced selection from the gold-standard dataset to augment the training set. This approach allowed the integration of artificially generated and gold data into two additional experimental groups: one trained with 3-shot (i.e., sentences generated from 3-shot setting) plus gold data, and another with 10-shot plus gold data. The data were converted to IOB format to train RoBERTa. We then compared the performance of these models to baseline scores from a model trained on fewer gold data sentences[8]. This comparison additionally aimed to evaluate the effectiveness of augmenting training sets with machine-generated data versus additional human-annotated data. We ensured consistency in hyperparameters and the number of training epochs to facilitate comparability[9]. Figure 4 shows an example of a few-shot learning.

5 Results

5.1 Experiment 1

We investigated the performance of supervised learning using gold-standard data applied with RoBERTa embeddings. The results, detailed in Table 2, highlight the best performance of the model trained using gold-standard data that includes both L1 and L2 annotations (Gold L1+L2 train model). It demonstrated the highest averaged F1 scores across all tested datasets: L1 (F1 = 0.912), L2 Written (F1 = 0.915), and L2 Spoken (F1 = 0.928). It also outperformed the other models in individual tag accuracy, securing the highest F1 scores for seven out of nine annotation types in both the L2 Written and Spoken datasets. Additionally, the

[7]The model was trained for only 400 iterations.

[8]This adjustment was made because the GPT-4 generated sentences typically had fewer ASC types, necessitating a reduction in the gold training data for a fair comparison.

[9]We used the same hyperparameter settings as the first experiment and also did the early stopping of stop at 400 iterations of the training data.

ASC	Silver train model			Gold L1 train model			Gold L1 + L2 train model		
	L1	L2Writ	L2Spok	L1	L2Writ	L2Spok	L1	L2Writ	L2Spok
ATTR	0.982	0.955	0.971	0.972	0.954	0.986	0.968	0.971	0.988
CAUS_MOT	0.794	0.764	0.690	0.818	0.833	0.710	0.857	0.867	0.710
DITRAN	0.757	0.862	1.000	0.919	0.914	0.842	0.865	0.881	0.947
INTRAN_MOT	0.763	0.755	0.774	0.800	0.770	0.789	0.772	0.807	0.843
INTRAN_RES	0.667	0.741	0.000	0.750	0.788	0.800	0.625	0.813	0.833
INTRAN_S	0.806	0.770	0.853	0.779	0.806	0.817	0.808	0.803	0.865
PASSIVE	0.932	0.865	0.875	0.920	0.775	0.938	0.940	0.865	0.909
TRAN_RES	0.853	0.714	0.588	0.884	0.800	0.625	0.881	0.792	0.625
TRAN_S	0.922	0.904	0.933	0.931	0.929	0.927	0.936	0.943	0.948
weightedAv	**0.902**	**0.885**	**0.907**	**0.908**	**0.900**	**0.905**	**0.912**	**0.915**	**0.928**

Table 2: F1-scores across ASC types, models, and registers, with the highest scores per tag in each dataset shaded (Experiment 1)

```
You are tasked with generating English sentences annotated
for argument structure constructions (ASCs). Each verb in the
sentence that forms an ASC should be labeled with one of
seven different ASC tags:
(1) transitive simple (TRAN_S),
(2) transitive resultative (TRAN_RES),
(3) caused-motion (CAUS_MOT),
(4) intransitive simple (INTRAN_S),
(5) intransitive motion (INTRAN_MOT),
(6) intransitive resultative (INTRAN_RES),
(7) ditransitive (DITRAN).

Requirements:
- Labels should be attached directly after the verb and
separated by a pipe '|' character. Words without an ASC tag
should be tagged with 'O'.
- Generate sentences that vary in complexity including
simple, compound, and complex structures to reflect realistic
and diverse usage, similar to examples below.
- Each sentence may contain multiple verbs and ASCs,
demonstrating a variety of grammatical constructs.
- Maintain grammatical correctness and logical coherence
throughout each sentence.

Follows are example annotations:
Bush|O nominated|TRAN_RES Jennifer|O M.|O Anderson|O for|O
a|O 15|O -|O year|O term|O
Today|O 's|O incident|O proves|TRAN_S that|O Sharon|O has|O
lost|TRAN_S his|O patience|O...
```

Figure 4: Example of prompting GPT-4 to generate ASC labels in a few-shot setting

model trained on the gold-standard L1 dataset (excluding L2) achieved top F1 scores for four out of nine tags in the L1 dataset. Overall, these results underscore the effectiveness of high-quality, manually annotated gold-standard data, as models trained on these datasets consistently outperformed those trained on silver-standard data in the development of effective ASC models.

5.2 Experiment 2

We explored prompt-guided annotation of unlabeled data using GPT-4. The results demonstrate that performance varied with the number of examples provided (Table 3). The zero-shot learning yielded the lowest F1 score at 0.434, while the 10-shot configuration showed an improvement, achieving the highest average F1 score of 0.631[10]. This indicates that more extensive example-driven guidance considerably enhances the model's effectiveness in automated ASC tagging tasks with GPT-4. However, the overall F1 scores were lower than the model trained solely on gold annotations (i.e., baseline[11]), and neither of the F1 scores for any ASC type exceeded those of the baseline model.

ASC tag (#)	zero-shot	3-shot	10-shot	baseline
CAUS_MOT (55)	0.121	0.446	0.483	0.907
DITRAN (46)	0.612	0.673	0.667	0.945
INTRAN_MOT (54)	0.562	0.674	0.684	0.825
INTRAN_RES (41)	0.130	0.525	0.730	0.822
INTRAN_S (105)	0.327	0.421	0.552	0.817
TRAN_RES (46)	0.213	0.306	0.485	0.863
TRAN_S (307)	0.676	0.700	0.742	0.922
weightedAv	0.434	0.563	0.631	0.888
Cost ($)	3.82	3.71	29.56	
Time (mins)	29	24	24	

Table 3: F1-scores for ASC tagging using GPT-4 (Experiment 2)

5.3 Experiment 3

We explore the use of prompt-guided generation of training data for training RoBERTa. The experiment was designed to first train the RoBERTa model using only the data generated by GPT-4 and then compare its performance with a model trained using gold standard data, as detailed in Table 4. The results reveal two key findings: First, increasing the number of examples, from 3-shot to 10-shot, enhanced model performance. The F1-scores generally improved with the number of examples provided, with the 10-shot configuration substantially outperforming the 3-shot across most categories.

[10]We additionally tested zero-shot learning by explicitly providing syntactic or semantic information about each construction to the model, but observed no improvement. Refer to Appendix C for detailed results and the prompts used.

[11]Among the three models in Experiment 1, the baseline model most closely resembles the best model trained on gold L1+L2 data, though it is not exactly the same, as it was only trained for 400 iterations.

This highlights the role of example-driven guidance in enhancing the quality of machine-generated training data; Second, despite the performance gains observed with an increased number of examples, models trained solely with gold data (gold1) consistently outperform those trained with the GPT-4 generated data (both 3-shot and 10-shot), particularly in more complex ASCs (e.g., CAUS_MOT, TRAN_RES). Although machine-generated data showed some potential in the training process for one ASC type (INTRAN_MOT), it still falls short of the effectiveness of human-annotated data.

Category	3-shot	10-shot	gold1
CAUS_MOT (55)	0.333	0.422	0.838
DITRAN (46)	0.367	0.632	0.867
INTRAN_MOT (54)	0.405	0.667	0.651
INTRAN_RES (41)	0.571	0.620	0.667
INTRAN_S (105)	0.303	0.485	0.742
TRAN_RES (46)	0.102	0.188	0.824
TRAN_S (307)	0.347	0.718	0.860
weightedAv	0.341	0.605	0.812
# of sentences	927	814	469
Cost ($)	3.31	6.59	
Time (mins)	18	20	

Table 4: Comparison of F1-scores for ASC tagging using different training sets, trained with RoBERTa (Experiment 3)

The second part of the experiment aimed to determine if augmenting the gold-standard training set with GPT-4-generated data could enhance the performance of the supervised learning model. As illustrated in Table 5, introducing machine-generated data (both 3-shot and 10-shot) into the gold data set did not consistently improve performance across all ASC tags[12]. The weighted average F1 score indicated that models trained with a combination of gold and machine-generated data (0.788 for 3-shot+gold and 0.808 for 10-shot+gold) generally performed less effectively or, at best, equally compared to those trained solely on gold-standard data (0.808).

Furthermore, the results demonstrate that the most significant improvement in performance was observed when gold data was augmented with additional gold data (gold1+gold2), achieving the highest weighted average F1 of 0.877. This underscores that while machine-generated data can improve training effectiveness for certain ASC types (e.g., TRAN_RES in 3-shot+gold, INTRAN_S in 3-shot+gold and 10-shot+gold), incorporating more human-annotated gold data (i.e., gold1+gold2) significantly enhances model accuracy across all investigated ASC categories. Upon closer examination of the machine-generated training data, it became evident that despite the prompts directing GPT-4 to generate sentences closely resembling the human-produced examples in the 10-shot set, the model struggled to capture the nuances present in sentences from human sources, such as the web corpus or L2 datasets (See Appendix D). In other words, GPT-4-generated sentences tend to be shorter and less complex, typically lacking multiple clauses, unlike the more elaborate sentences crafted by humans. This limitation likely impacted the quality of the training data and, consequently, the effectiveness of the training outcomes.

ASC tag (#)	gold1	gold1 +3-shot	gold1 +10-shot	gold1 +gold2
CAUS_MOT (55)	0.838	0.731	0.782	0.914
DITRAN (46)	0.867	0.756	0.824	0.920
INTRAN_MOT (54)	0.651	0.644	0.727	0.814
INTRAN_RES (41)	0.667	0.615	0.695	0.831
INTRAN_S (105)	0.742	0.760	0.751	0.816
TRAN_RES (46)	0.824	0.857	0.782	0.886
TRAN_S (307)	0.860	0.851	0.863	0.900
weightedAv	0.808	0.788	0.808	0.877
# of trained sentences	469	1396	1283	938

Table 5: Comparison of F1-scores for ASC tagging using different training sets – combined with the gold-standard data, trained with RoBERTa (Experiment 3)

6 Conclusions

This study highlights the potential of integrating PLMs into linguistic analysis frameworks, particularly for examining the characteristics of ASCs in the context of modeling L1 and L2 learning/development. RoBERTa, when trained on gold-standard datasets, demonstrated the best performance in ASC annotations, underscoring the importance of comprehensive, high-quality annotated data. Additionally, the use of GPT-4 for prompt-guided annotation and data generation offered some insights into the effectiveness of synthetic data in model training. While these methods did not surpass the F1 scores of the baseline model trained solely on gold-standard annotations, they proved effective in identifying and processing certain types of ASCs.

Future directions This study serves as a promising foundation for automated annotation systems in

[12]There are some cases where it slightly enhances the model's effectiveness, as seen in the TRAN_RES and INTRAN_S tags.

both L1 and L2 language contexts. However, it did not directly assess the effectiveness of ASC annotation in automatic writing evaluation or feedback systems, which represent critical avenues for future research and applications of NLP for building educational applications.

Limitations

The accuracy of ASC annotation was assessed across three linguistic domains—L1 written, L2 written, and L2 spoken—but only a single register within each domain was examined in Experiment 1. Experiments 2 and 3 did not comprehensively explore model performance across different domains. Consequently, the applicability of these models in other registers, such as L2 written narratives or L2 argumentative speeches, remains uncertain, particularly with the RoBERTa model. Furthermore, the GPT-4 model should have also included investigations into two additional ASC types (PASSIVE, ATTRIBUTE) and comparisons across different linguistic domains. Additionally, due to the limited scope of the L2 datasets, certain ASC types, such as transitive and intransitive resultative constructions, were underrepresented in the test sets. Therefore, the annotation accuracy for these specific ASCs should be interpreted with caution.

Supplementary Materials

All data and models are available in `https://github.com/LCR-ADS-Lab/ASC-Treebank`. All contributions are licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (CC BY-NC-SA 4.0).

References

Alan Akbik, Laura Chiticariu, Marina Danilevsky, Yunyao Li, Shivakumar Vaithyanathan, and Huaiyu Zhu. 2015. Generating high quality proposition banks for multilingual semantic role labeling. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 397–407.

Ben Ambridge, Julian M Pine, Caroline F Rowland, Franklin Chang, and Amy Bidgood. 2013. The retreat from overgeneralization in child language acquisition: Word learning, morphology, and verb argument structure. *Wiley Interdisciplinary Reviews: Cognitive Science*, 4(1):47–62.

Øistein E Andersen, Julien Nioche, Ted Briscoe, and John Carroll. 2008. The bnc parsed with rasp4uima. In *LREC*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

Giulia ML Bencini and Adele E Goldberg. 2000. The contribution of argument structure constructions to sentence meaning. *Journal of Memory and Language*, 43(4):640–651.

Yevgeni Berzak, Jessica Kenney, Carolyn Spadine, Jing Xian Wang, Lucia Lam, Keiko Sophie Mori, Sebastian Garza, and Boris Katz. 2016. Universal dependencies for learner english. *arXiv preprint arXiv:1605.04278*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Marie-Catherine de Marneffe and Christopher D Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*, pages 1–8.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bosheng Ding, Chengwei Qin, Linlin Liu, Yew Ken Chia, Shafiq Joty, Boyang Li, and Lidong Bing. 2022. Is gpt-3 a good data annotator? *arXiv preprint arXiv:2212.10450*.

Masaki Eguchi and Kristopher Kyle. 2023. Span identification of epistemic stance-taking in academic written english. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 429–442. Association for Computational Linguistics (ACL).

Nick C Ellis. 2002. Frequency effects in language processing: A review with implications for theories of implicit and explicit language acquisition. *Studies in second language acquisition*, 24(2):143–188.

Nick C Ellis and Fernando Ferreira-Junior. 2009. Construction learning as a function of frequency, frequency distribution, and function. *The Modern language journal*, 93(3):370–385.

Allyson Ettinger, Jena D Hwang, Valentina Pyatkin, Chandra Bhagavatula, and Yejin Choi. 2023. "you are an expert linguistic annotator": Limits of llms as analyzers of abstract meaning representation. *arXiv preprint arXiv:2310.17793*.

Charles J. Fillmore. 1968. Lexical entries for verbs. *Foundations of language*, pages 373–393.

Charles J Fillmore, Christopher R Johnson, and Miriam RL Petruck. 2003. Background to framenet. *International journal of lexicography*, 16(3):235–250.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Adele E Goldberg. 1995. *Constructions: A construction grammar approach to argument structure*. University of Chicago Press.

Adele E Goldberg. 2003. Constructions: A new theoretical approach to language. *Trends in cognitive sciences*, 7(5):219–224.

Adele E Goldberg, Devin M Casenhiser, and Nitya Sethuraman. 2004. Learning argument structure generalizations.

Stefan Th Gries and Stefanie Wulff. 2005. Do foreign language learners also have constructions? *Annual review of cognitive linguistics*, 3(1):182–200.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, Adriane Boyd, et al. 2020. spacy: Industrial-strength natural language processing in python.

Haerim Hwang and Hyunwoo Kim. 2023. Automatic analysis of constructional diversity as a predictor of efl students' writing proficiency. *Applied Linguistics*, 44(1):127–147.

Daniel Jurafsky and James H Martin. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.

Kristopher Kyle. 2016. *Measuring syntactic development in L2 writing: Fine grained indices of syntactic complexity and usage-based indices of syntactic sophistication*. Ph.D. thesis, Georgia State University, Atlanta, GA.

Kristopher. Kyle and Scott Crossley. 2017. Assessing syntactic sophistication in l2 writing: A usage-based approach. *Language Testing*, 34(4):513–535.

Kristopher Kyle, Masaki Eguchi, Aaron Miller, and Theodore Sither. 2022. A dependency treebank of spoken second language english. In *Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2022)*, pages 39–45.

Kristopher Kyle and Hakyung Sung. 2023. An argument structure construction treebank. In *Proceedings of the First International Workshop on Construction Grammars and NLP (CxGs+ NLP, GURT/SyntaxFest 2023)*, pages 51–62.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Alessio Miaschi and Felice Dell'Orletta. 2020. Contextual and non-contextual word embeddings: an in-depth linguistic investigation. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 110–119.

Anat Ninio. 1999. Pathbreaking verbs in syntactic development and the question of prototypical transitivity. *Journal of child language*, 26(3):619–653.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal dependencies v2: An evergrowing multilingual treebank collection. *arXiv preprint arXiv:2004.10643*.

Mary Catherine O'Connor and Paul Kay. 2003. Regularity and idiomaticity in grammatical constructions: The case of let alone. In *The new psychology of language*, pages 243–270. Psychology Press.

Matthew O'Donnell and Nick Ellis. 2010. Towards an inventory of english verb argument constructions. In *Proceedings of the NAACL HLT Workshop on Extracting and Using Constructions in Computational Linguistics*, pages 9–16.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Malka Rappaport Hovav and Beth Levin. 1998. Building verb meanings. *The projection of arguments: Lexical and compositional factors*, pages 97–134.

Ute Römer, Audrey Roberson, Matthew B O'Donnell, and Nick C Ellis. 2014. Linking learner corpus and experimental data in studying second language learners' knowledge of verb-argument constructions. *ICAME Journal*, 38(1):115–135.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Natalia Silveira, Timothy Dozat, Marie-Catherine De Marneffe, Samuel R Bowman, Miriam Connor, John Bauer, and Christopher D Manning. 2014. A gold standard dependency corpus for english. In *LREC*, pages 2897–2904. Citeseer.

Hakyung Sung and Kristopher Kyle. 2024. Annotation scheme for English argument structure constructions treebank. In *Proceedings of The 18th Linguistic Annotation Workshop (LAW-XVIII)*, pages 12–18, St. Julians, Malta. Association for Computational Linguistics (ACL).

Michael Tomasello. 2005. *Constructing a language: A usage-based theory of language acquisition*. Harvard university press.

Michael Tomasello and Patricia J. Brooks. 1998. Young children's earliest transitive and intransitive constructions. *Cognitive Linguistics*, 9(4):379–396.

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.

Danni Yu, Luyang Li, Hang Su, and Matteo Fuoli. 2023. Assessing the potential of llm-assisted annotation for corpus-based pragmatics and discourse analysis.

A Distribution of ASC types and tokens

	L1			L2 Written			L2 Spoken		
ASC	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test
ATTR	2,058	258	223	399	445	445	242	266	252
CAUS_MOT	641	61	64	26	30	31	18	18	17
DITRAN	235	31	19	59	47	54	16	12	9
INTRAN_MOT	502	55	50	86	79	85	91	82	67
INTRAN_RES	172	23	18	15	13	16	11	5	7
INTRAN_S	1,154	135	106	243	209	210	146	190	189
PASSIVE	867	102	89	72	59	73	16	18	16
TRAN_RES	622	77	64	25	28	23	6	5	5
TRAN_S	4,900	598	596	858	824	806	506	426	453
Total	11,151	1,340	1,229	1,783	1,734	1,743	1,052	1,022	1,015

B Hyperparameter values

Hyperparameter	**Selected**
Num hidden units	200
Embedded vector space	50
Number of layers	1
Dropout rate of layers	0.5
Beam size	1
Attention type	soft
Optimization algorithm	Adam
Learning rate	0.001
Num epoch	60
Batch size	50
Max grad norm	5.0
GPU allocator	pytorch
Seed	0
Batch size	128
Hidden width	64
Maxout pieces	2
Max batch items	4096
Transformer model name	roberta-base
Window size (get spans)	128
Stride (get spans)	96
Accumulate gradient	3
Dropout (training)	0.1
Patience	1600
Max epochs	0
Max steps	20000
Eval frequency	200
Initial learning rate	0.00005
Warmup steps	250
Total steps (learn rate)	20000
L2 regularization	0.01
Grad clip	1.0
Epsilon (optimizer)	0.00000001

C Experiment 2: Additional experiments

Table 6 and Figures 5 & 6 below present a comparison of F1-scores for ASCs tagging using GPT-4 in zero-shot learning scenarios. These scenarios vary by the inclusion of either syntactic (syn) or semantic (sem) descriptions accompanying each construction type.

Category (#)	zero-shot	zero-shot+syn desc	zero-shot+sem desc
CAUS_MOT (55)	0.121	0.206	0.225
DITRAN (46)	0.612	0.436	0.512
INTRAN_MOT (54)	0.562	0.584	0.552
INTRAN_RES (41)	0.130	0.118	0.125
INTRAN_S (105)	0.327	0.239	0.248
TRAN_RES (46)	0.213	0.148	0.290
TRAN_S (307)	0.676	0.619	0.629
weightedAV (654)	0.435	0.389	0.416
Cost ($)	3.82	10.74	4.79
Time (mins)	29	27	24

Table 6: F1-Scores for ASC tagging using GPT-4 in zero-shot learning, compared with when the prompt included the syntactic (syn) or semantic (sem) descriptions of each construction type.

```
You are tasked with annotating English argument structure constructions
(ASCs) in sentences.
Your role is to assign one of seven different ASC labels to verbs that
form ASCs in each sentence:
(1) transitive simple: subject-verb-object (TRAN_S),
(2) transitive resultative: subject-verb-object-complement (TRAN_RES),
(3) caused-motion: subject-verb-object-oblique/directional particle
(CAUS_MOT),
(4) intransitive simple: subject-verb (INTRAN_S),
(5) intransitive motion: subject-verb-oblique/directional particle
(INTRAN_MOT),
(6) intransitive resultative: subject-verb-complement (INTRAN_RES),
(7) ditransitive: subject-verb-indirect object-direct object (DITRAN)
Tag only the labels. Note that a sentence may contain multiple ASCs.
Please format your annotations in a simplified CoNLL-U format. For each
word in the sentence, list its order number, the word itself, and its
label if it is a verb participating in an ASC; otherwise, use an
underscore (_). Example:
# text = John throws the ball.
1\tJohn\t_
2\tthrows\t[tag a label]
3\tthe\t_
4\tball\t_
5\t.\t_

Please annotate the following sentence:
```

Figure 5: Prompt for ASC tagging using syntactic (syn) descriptions in zero-shot learning scenarios.

```
You are tasked with annotating English argument structure constructions
(ASCs) in sentences.
Your role is to assign one of seven different ASC labels to verbs that
form ASCs in each sentence:
(1) transitive simple: agent-verb-theme/patient (TRAN_S),
(2) transitive resultative: agent-verb-theme/patient-result (TRAN_RES),
(3) caused-motion: agent-verb-theme/patient-goal (CAUS_MOT),
(4) intransitive simple: theme/patient-verb (INTRAN_S),
(5) intransitive motion: agent-verb-goal (INTRAN_MOT),
(6) intransitive resultative: agent-verb-result (INTRAN_RES),
(7) ditransitive: agent-verb-recipient-theme/patient (DITRAN)
Tag only the labels. Note that a sentence may contain multiple ASCs.
Please format your annotations in a simplified CoNLL-U format. For each
word in the sentence, list its order number, the word itself, and its
label if it is a verb participating in an ASC; otherwise, use an
underscore (_). Example:
# text = John throws the ball.
1\tJohn\t_
2\tthrows\t[tag a label]
3\tthe\t_
4\tball\t_
5\t.\t_

Please annotate the following sentence:
```

Figure 6: Prompt for ASC tagging using semantic descriptions (sem) in zero-shot learning scenarios.

D Experiment 3: Comparison of human and GPT-4 generated sentences

Table D presents examples of sentences randomly extracted from the gold dataset used in Experiment 3, alongside sentences generated by the GPT-4 model under a 10-shot setting.

Human-generated Sentences
1. What if Google expanded on its search-engine (and now e-mail) wares into a full-fledged operating system?
2. I doubt the very few who actually read my blog have not come across this yet, but I figured I would put it out there anyways.
3. Click here to view it.
4. One of the pictures shows a flag that was found in Fallujah.
5. Compare the flags to the Fallujah one.
6. Let me join the chorus of annoyance over Google's new toolbar, which, as noted in the linked article, commits just about every sin an online marketer could commit, and makes up a few new ones besides.
7. You don't need to use their site, you can opt-out of sharing your information, you don't need to send stuff to anyone with a Gmail account, and if – wonder of wonders – you're worried that you might send something to someone who would forward an excerpt to someone who would then store it on a Gmail account... you have far, far too much time on your hands.
8. On the other hand, it looks pretty cool.
9. Keep his cage open and go on your computer, or read a book, etc and maybe he will come out to you.
10. Please let us know if you need anything else.
GPT-4 generated Sentences
1. I ran to the store to buy milk.
2. She was so excited to see her friends at the party.
3. The sun set, painting the sky with hues of orange and pink.
4. He gave me the book that I wanted.
5. The leaves fell from the trees as autumn arrived.
6. She baked the cake until it was golden brown.
7. He threw the ball to his dog.
8. The dog chased the cat up the tree.
9. The bird flew out of the cage.
10. The child threw the ball into the hoop.