Blending Dependency Parsers With Language Models

Nicos Isaak[a]
Computational Cognition Lab, Cyprus

Keywords: Language Models, Deep Learrning, Natural Language Processing, Dependency Parsing, Parsers.

Abstract: Recent advances in the AI field triggered an increasing interest in tackling various NLP tasks with language models. Experiments on several benchmarks demonstrated their effectiveness, potential, and role as a central component in modern AI systems. However, when training data are limited, specialized expertise is needed in order to help them perform complex kind-of-reasoning and achieve better results. It seems that extensive experiments with additional semantic analysis and fine-tuning are needed to achieve improvements on downstream NLP tasks. To address this complex problem of achieving better results with language models when training data are limited, we present a simplified way that automatically improves their learned representations with extra-linguistic knowledge. To this end, we show that further fine-tuning with semantics from state-of-the-art dependency parsers improves existing language models on specialized downstream tasks. Experiments on benchmark datasets we undertook show that the blending of language models with dependency parsers is promising for achieving better results.

1 INTRODUCTION

In recent years, language models have become increasingly important to the AI research community. The AI field has been revolutionized by language models at scale mostly based on the Transformer's architecture (Vaswani et al., 2017), which is considered the crème de la crème of the Machine Learning community. Moreover, the number of papers involving solutions with language models has been increasing rapidly, suggesting that language models could be considered the new silver bullet of the ML community (Devlin et al., 2018; Liu et al., 2019; Radford et al., 2019; Brown et al., 2020; He et al., 2020).

Language model parameters seem to store a form of knowledge that can help them tackle various NLP tasks (Wang et al., 2020). The aim is to utilize their densely connected collection of inferential knowledge for question answering, pronoun resolution, text summarization, token classification, or text similarity tasks. However, when data are limited, further fine-grained strategies are usually needed involving new training data to help them tackle various downstream tasks. Typical approaches involve the utilization of exemplars demonstrating the task (Brown et al., 2020) or the fine-tuning with more training data as a simplified form of adding extra knowledge.

[a] https://orcid.org/0000-0003-2353-2192

Given that language model commonsense-and-reasoning abilities are more accurate when employed with semantic relations, something dependency parsers do really well (Isaak and Michael, 2016), why not just directly inject them with all the semantic or syntactic knowledge of the modern parsers? Moreover, this will add to the well-known notion that neural nets have been implemented to reduce feature engineering, as this fine-grained strategy does not require the building of complex relations in order to enhance each model's prediction abilities.

Consequently, here we employ a simplified blending mechanism to enhance language models' knowledge capacity via dependency parsers. We know that dependency parsers are important as they have been extensively utilized, mainly in classic AI research fields. Since the early days of AI (McCarthy et al., 2006), researchers have increasingly incorporated them in AI systems to tackle various NLP tasks and challenges, such as the Turing Test and the Winograd Schema Challenge (Isaak and Michael, 2017, 2016; Sharma et al., 2015; Michael, 2013; Rahman and Ng, 2012).

Traditional rule-based systems tackle various tasks by employing strategies incorporating the output of dependency parsers. From this standpoint, parsers might augment language models' capabilities by limiting the situations they have never met in train-

Isaak, N.
Blending Dependency Parsers With Language Models.
DOI: 10.5220/0011781800003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 3*, pages 813-820
ISBN: 978-989-758-623-1; ISSN: 2184-433X

ing. For instance, we know that, based on dependency parsers, each word found in a sentence is connected to other words via binary relations between superior and inferior words (see Figure 1) (Kübler et al., 2009). We suspect that fine-tuning language models with knowledge found in dependency parsers will help them achieve better accuracy on NLP tasks or even help them in a zero-shot setting, that is, learning to classify classes a model has never seen before (Xian et al., 2017; Romera-Paredes and Torr, 2015; Wei et al., 2021).

In this paper, we employ a simplified way of enhancing language models with semantics from the spaCy dependency parser[1], which is one of the fastest parsers in the NLP community. In this regard, each language model training example is turned into a densely connected network of schemas based on semantic relations found in English sentences and phrases (Isaak and Michael, 2016; Michael, 2013).

We start by presenting what language models are all about. Next, we put forward our simplified methodology of enhancing language models training material and we go on by presenting our empirical evaluation results against the Yelp Review Dataset, a multi-label classification task that refers to the sentiment analysis of several reviews.

The results show that dependency parsers and language models blend well, at least in small chunks of the Yelp Review Dataset, showing that injecting training material with semantic information leads to performance improvements in accuracy. Testing if our method could have the same results when applied to large training datasets is out of the reach of this paper —it is already proven that having access to large training data could lead models to achieve high-accuracy results. Finally, our work does not purport to replace but only to add to current data augmenting procedures the research community is already using (Isaak and Michael, 2021).

2 LANGUAGE MODELS

According to Bengio (2008), language models are algorithms able to capture the statistical characteristics of sequences of words in written language, allowing them to make predictions for missing words or following sequences of text given a previous form of text. These models are primarily trained on enormous amounts of text, sometimes at a petabyte scale, which can be easily fine-tuned on downstream tasks to maximize their performance.

[1]https://spacy.io

Models such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), GPT-2 (Radford et al., 2019), GPT-3 (Brown et al., 2020), DeBERTa (He et al., 2020), have revolutionized the NLP field. By processing millions of data points, these models make it possible for machines to analyze or interpret bodies of text as they analyze the pattern of human language to predict. As they are designed to represent a language domain, language models can write poems or songs or even perform sentiment analysis, sometimes better than humans.

Learning through training examples allows a language model to transform and represent words based on a feature-vector representation. In this sense, each direction in the feature-vector space refers to a semantic characteristic of words in that similar words tend to be next to each other along some directions. This allows the model to replace similar words in the same context and to make predictions for words not seen in the training samples. In this regard, we can have large models with billions of parameters that can be either autoregressive, like GPT-3, or able to predict sequences of sentences or a missing token from a sentence, like BERT.

In the last decade, remarkable achievements have been made as they have significantly boosted the accuracy of many NLP tasks, as single models can be fine-tuned and utilized in various tasks without substantial task-specific architecture modifications. These models have proven flexible as they can tackle numerous NLP problems that classic AI has struggled with for decades. Specifically, these models achieved breakthrough performance as they helped mitigate challenges such as named-entity recognition, question answering, sentence completion, machine translation, summarization, information retrieval, and sentiment analysis.

However, because of memory limitations of available hardware, a topic of interest has been the discovery of creative ways to lighten models while maintaining high-performance results (Lan et al., 2019). In this sense, more efficient knowledge extraction techniques are needed in order to achieve more robust performance with cheaper models (Raffel et al., 2020).

3 DEPENDENCY PARSERS

Dependency parsers analyze text structures (e.g., sentences, paragraphs) to discover the syntactic dependencies on a word level, that is, binary asymmetric relations the words are linked (Kübler et al., 2009; Rasooli and Tetreault, 2015). As stated by Kübler et al. (2009), each word found in a sentence is de facto

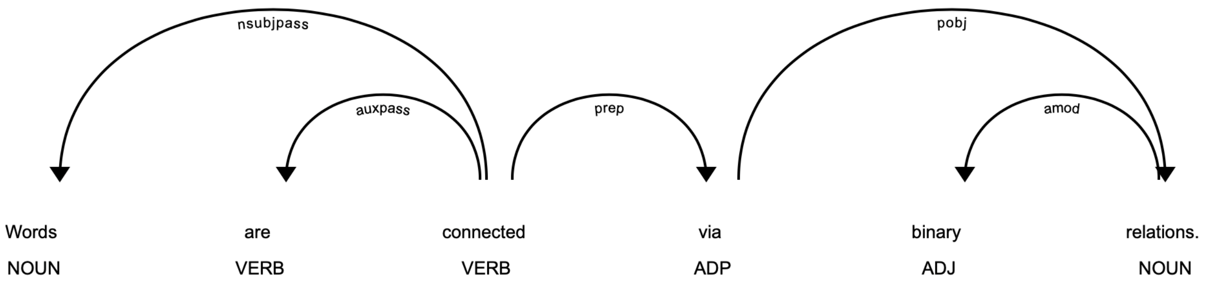


Figure 1: An example of a Dependency Parser Output: Parsers are able to analyze sentences and display binary relations between words.

connected to other words in a way that not only is it not isolated, but it plays a significant role in the sentence structure. Moreover, each connection of words refers to binary relations between superior and inferior words, that is, words that have grammatical function based on other words in a sentence (Nivre, 2005).

To illustrate, we can have relations such as nsubj (cat, plays) and dobj (plays, ball), where the former refers to the nominal subject and the latter to the direct object of a sentence (see Table 1). Binary relations can be further combined to develop enhanced relationships, called scenes or triples, such as cat (1), ball (2), plays (1, 2), indirectly telling us that a *cat is playing with a ball* (Isaak and Michael, 2016; Michael, 2013). Furthermore, through scenes, we can build simple sentences by combining the words participating in the triple relations into simplified English sentences. For instance, from the scenes given above (cat (1), ball (2), plays (1, 2)), we can form a simplified sentence such as *A cat plays with a ball* (Michael, 2013).

With scenes like the above, we can tap into the richness of knowledge and reasoning areas to tackle challenges like the Winograd Schema Challenge (Levesque, 2014; Isaak and Michael, 2016). In this sense, blending language models with dependency parsers could lead to the development of enhanced models able to find extra convoluted relationships between words in sentences, to tackle more NLP tasks.

4 METHODOLOGY

In this section, we examine if additional fine-tuning with semantic scenes could lead language models to achieve better results. According to Radford et al. (2019), when trained on large or a variety of datasets, language models start learning various relations between sequences of text without requiring explicit supervision.

For the purpose of this study, we formulate the following null hypothesis: “Additional fine-tuning with scenes built from dependency parsers does not affect existing model performance on examined NLP tasks”. As a result, we examine if pre-trained language models can presumably untangle and extract relations from dependency parsers to outperform their previous results.

Table 1: Stanford and spaCy Parser Output: For the sentence, *The cat caught the mouse because it was clever*. In the first line, we see each word’s part of speech, and in the second line, the typed dependencies —example is taken from Isaak and Michael (2016).

POS	The/DT cat/NN caught/VBD the/DT mouse/NN because/IN it/PRP was/VBD clever/JJ
Stanford	det(cat-2, The-1)
	nsubj(caught-3, cat-2)
	root(ROOT-0, caught-3)
	det(mouse-5, the-4)
	dobj(caught-3, mouse-5)
	mark(clever-9, because-6)
	nsubj(clever-9, it-7)
	cop(clever-9, was-8)
	advcl(caught-3, clever-9)
spaCy	The det
	cat nsubj
	caught ROOT
	The det
	mouse dobj
	because mark
	it nsubj
	was advcl
	clever acomp

Below, we illustrate how language model results can be enhanced through the densely connected relations of dependency parsers. In particular, we will show how spaCy’s dependencies are utilized through language models on specified downstream tasks.

For testing purposes, our downstream task refers to the yelp_review_full 2015 dataset, which can be easily downloaded from the Hugging Face platform[2]. The dataset is mainly utilized in classification tasks, where, given the text, we have to predict the sentiment, which is a value of 1 to 5 (see Table 2). We can

[2]https://huggingface.co/datasets/yelp_review_full

find more about the dataset in the paper where it was introduced (Zhang et al., 2015)

Table 2: A snapshot of two training examples from the yelp-review-full dataset: Given the text, we have to estimate a label that is a value in the range of 0 to 4. The values refer to star review values of one to five stars, accordingly.

Yelp-review dataset (2015)	
{ 'label': 4, 'text': Top notch doctor in a top notch practice. Can't say I am surprised when I was referred to him ...}	5 stars
{ 'label': 0, 'text': We went on a weeknight. Place was not busy waited over 20 minutes for drinks and to have our order taken...}	0 stars

During this preliminary stage, we started experimenting with the well-known BERT-base model (bert-base-cased) for testing purposes. According to Devlin et al. (2018), in Bert-like models, there is minimal difference between the pre-training and fine-tuning phases on downstream tasks. In this sense, the model can be easily fine-tuned on various downstream tasks via the utilization of the same pre-trained parameters. To illustrate, our examined model is initialized with their pre-trained parameters and then fine-tuned using labeled data from the yelp_review dataset.

Based on the Bert-base LM and the Yelp-Dataset, we examine the effects that the dependency scenes from spaCy might have on their performance. Given that most language models are trained for specified tasks and objectives, the blending with spaCy dependencies might lead to better results without fine-tuning with many data or epochs.

Additionally, to show the key findings of the blending mechanism, the model is tested on a minimum amount of training and testing material. Recall that our method refers to testing on small training datasets, meaning cases, our models cannot find enough relations between the words of sentences to tackle the required task effectively. Testing if our method could have the same results when applied to large training datasets is out of the reach of this paper —it is already proven that having access to large training data could lead to high-accuracy results.

As depicted in Figure 2, we build the whole procedure around a unified-platform design, which can handle various datasets and models. The platform offers simplicity because anyone can easily understand the flow of parsed data between models and the utilized parsers, integrability, that is, new models can be easily added to be tested, and versatility, based on the variety of tasks it is built upon.

In its current form, the platform is given access to training examples, such as the Yelp Review Dataset, and a predefined model to work with (see Figure 2) — the platform requires each training material to be an English phrase or sentence. In the first step, through spaCy, each acquired phrase is split into sentences, based on which the parsing phase starts. Next, for each sentence, the *Scene Constructor* component is called to develop first-order semantic scenes (triples), which are logical formulae similar to Prolog rules (see steps three and four in Figure 2). Afterward, the *Sentence Builder* is called to build sentences in a simplified form —to keep things simple, emphasis was given on constructing sentences in the form of *The "word1" is related to "word2" through "relation"*. Given that language models already contain a densely connected knowledge of relations, emphasis was given on constructing sentences for triples in which at least one word is not presented in the tokenizer's vocabulary. If no triple exists for any given sentence, no additional sentences are produced. Otherwise, through the *Blending Mechanism* component, each produced sentence is added to the initial training example.

Each round was tested with (Blended approach) and without our methodology (Vanilla approach). Finally, our language model was downloaded from HuggingFace[3], and all our experiments were run on a subscribed version of the Gradient Platform[4]. The experiments ran for several weeks and yielded an average of 1389 semantic scenes (triples) for each round.

To keep things simple, via a batch size of 8, we fine-tune for one epoch over our dataset with a learning rate of 5e-5. Then, to compare our results, we start by fine-tuning the models on specified training and testing data and then repeating the same experiments with the enhanced datasets, that is, the dataset produced via our Blended approach. To evaluate how the scenes from dependency parsers affect the performance of an existing model, we tested our methodology with varying training materials (see Figure 2). In particular, at first, we run the vanilla approach in ten rounds (two circles of five rounds each) with varying values of a parameter S that specifies which training data are selected. In particular, in each round, chunks of 1000 randomly selected training reviews (see Table 2) are selected and tested in two chunks of the testing dataset. To compare our results, in the first five rounds, we tested the vanilla approach on the first 300 reviews, and in the second, we repeated the same procedure on the following 300 reviews. Afterward, we repeated the same experiments under the Blended approach, that is, prior to the training process of the language model, each training sample is being parsed and enhanced with additional sentences (see Figure 2).

The variety of selected datasets for both approaches was considered to show the effectiveness

[3]https://huggingface.co/models

[4]https://www.paperspace.com/gradient

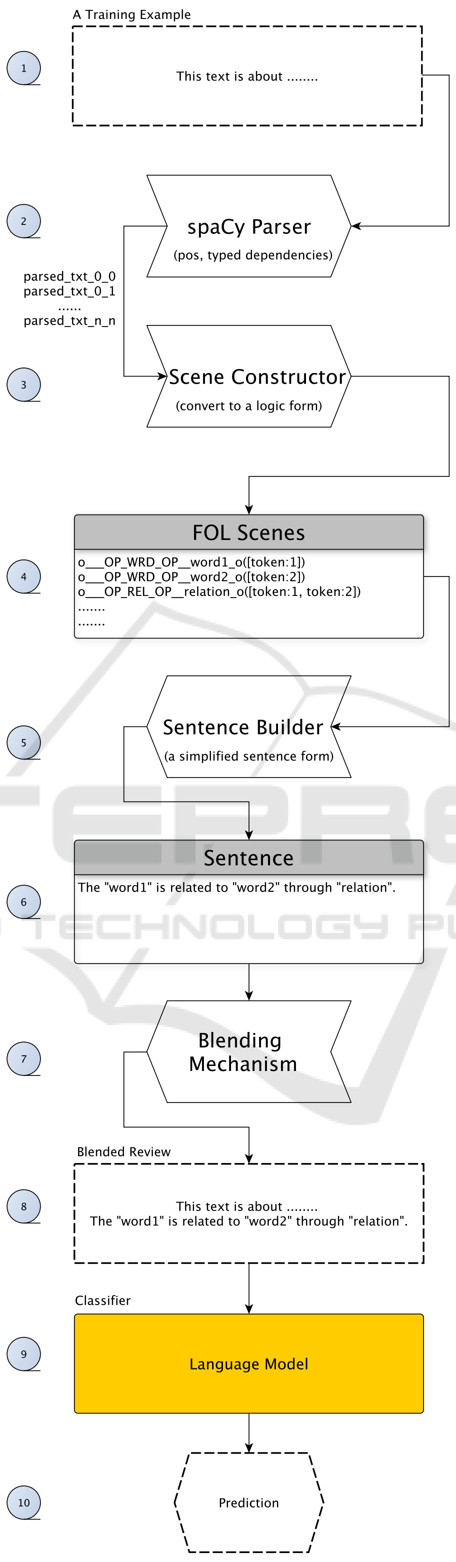
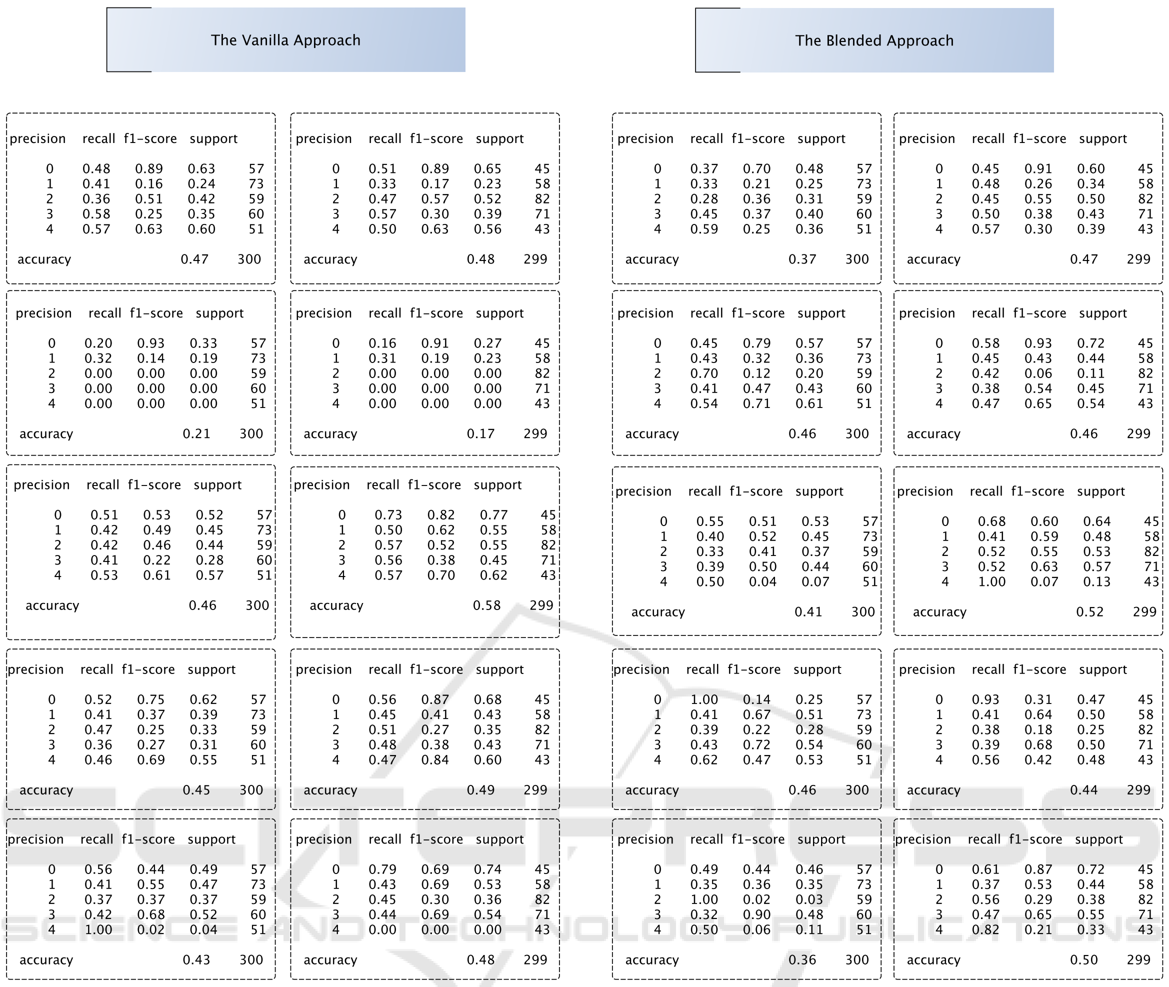

A Training Example
1
This text is about ....
2
spaCy Parser
(pos, typed dependencies)
parsed_txt_0_0
parsed_txt_0_1
......
parsed_txt_n_n
3
Scene Constructor
(convert to a logic form)
FOL Scenes
4
o___OP_WRD_OP__word1_o([token:1])
o___OP_WRD_OP__word2_o([token:2])
o___OP_REL_OP__relation_o([token:1, token:2])
....
....
5
Sentence Builder
(a simplified sentence form)
Sentence
6
The "word1" is related to "word2" through "relation".
7
Blending Mechanism
Blended Review
8
This text is about ....
The "word1" is related to "word2" through "relation".
Classifier
9
Language Model
10
Prediction

Figure 2: Our Blending Mechanism: For a given phrase, i) the spaCy parser along the Scene Constructor build the scenes for each sentence, ii) based on the developed scenes, the sentence-builder designs new sentences, and iii) the blending mechanism enhances the given phrase with the newly developed sentences.



The Vanilla Approach

```
precision    recall  f1-score   support

        0      0.48      0.89      0.63        57
        1      0.41      0.16      0.24        73
        2      0.36      0.51      0.42        59
        3      0.58      0.25      0.35        60
        4      0.57      0.63      0.60        51

 accuracy                          0.47       300
```

```
precision    recall  f1-score   support

        0      0.51      0.89      0.65        45
        1      0.33      0.17      0.23        58
        2      0.47      0.57      0.52        82
        3      0.57      0.30      0.39        71
        4      0.50      0.63      0.56        43

 accuracy                          0.48       299
```

```
precision    recall  f1-score   support

        0      0.20      0.93      0.33        57
        1      0.32      0.14      0.19        73
        2      0.00      0.00      0.00        59
        3      0.00      0.00      0.00        60
        4      0.00      0.00      0.00        51

 accuracy                          0.21       300
```

```
precision    recall  f1-score   support

        0      0.16      0.91      0.27        45
        1      0.31      0.19      0.23        58
        2      0.00      0.00      0.00        82
        3      0.00      0.00      0.00        71
        4      0.00      0.00      0.00        43

 accuracy                          0.17       299
```

```
precision    recall  f1-score   support

        0      0.51      0.53      0.52        57
        1      0.42      0.49      0.45        73
        2      0.42      0.46      0.44        59
        3      0.41      0.22      0.28        60
        4      0.53      0.61      0.57        51

 accuracy                          0.46       300
```

```
precision    recall  f1-score   support

        0      0.73      0.82      0.77        45
        1      0.50      0.62      0.55        58
        2      0.57      0.52      0.55        82
        3      0.56      0.38      0.45        71
        4      0.57      0.70      0.62        43

 accuracy                          0.58       299
```

```
precision    recall  f1-score   support

        0      0.52      0.75      0.62        57
        1      0.41      0.37      0.39        73
        2      0.47      0.25      0.33        59
        3      0.36      0.27      0.31        60
        4      0.46      0.69      0.55        51

 accuracy                          0.45       300
```

```
precision    recall  f1-score   support

        0      0.56      0.87      0.68        45
        1      0.45      0.41      0.43        58
        2      0.51      0.27      0.35        82
        3      0.48      0.38      0.43        71
        4      0.47      0.84      0.60        43

 accuracy                          0.49       299
```

```
precision    recall  f1-score   support

        0      0.56      0.44      0.49        57
        1      0.41      0.55      0.47        73
        2      0.37      0.37      0.37        59
        3      0.42      0.68      0.52        60
        4      1.00      0.02      0.04        51

 accuracy                          0.43       300
```

```
precision    recall  f1-score   support

        0      0.79      0.69      0.74        45
        1      0.43      0.69      0.53        58
        2      0.45      0.30      0.36        82
        3      0.44      0.69      0.54        71
        4      0.00      0.00      0.00        43

 accuracy                          0.48       299
```

The Blended Approach

```
precision    recall  f1-score   support

        0      0.37      0.70      0.48        57
        1      0.33      0.21      0.25        73
        2      0.28      0.36      0.31        59
        3      0.45      0.37      0.40        60
        4      0.59      0.25      0.36        51

 accuracy                          0.37       300
```

```
precision    recall  f1-score   support

        0      0.45      0.91      0.60        45
        1      0.48      0.26      0.34        58
        2      0.45      0.55      0.50        82
        3      0.50      0.38      0.43        71
        4      0.57      0.30      0.39        43

 accuracy                          0.47       299
```

```
precision    recall  f1-score   support

        0      0.45      0.79      0.57        57
        1      0.43      0.32      0.36        73
        2      0.70      0.12      0.20        59
        3      0.41      0.47      0.43        60
        4      0.54      0.71      0.61        51

 accuracy                          0.46       300
```

```
precision    recall  f1-score   support

        0      0.58      0.93      0.72        45
        1      0.45      0.43      0.44        58
        2      0.42      0.06      0.11        82
        3      0.38      0.54      0.45        71
        4      0.47      0.65      0.54        43

 accuracy                          0.46       299
```

```
precision    recall  f1-score   support

        0      0.55      0.51      0.53        57
        1      0.40      0.52      0.45        73
        2      0.33      0.41      0.37        59
        3      0.39      0.50      0.44        60
        4      0.50      0.04      0.07        51

 accuracy                          0.41       300
```

```
precision    recall  f1-score   support

        0      0.68      0.60      0.64        45
        1      0.41      0.59      0.48        58
        2      0.52      0.55      0.53        82
        3      0.52      0.63      0.57        71
        4      1.00      0.07      0.13        43

 accuracy                          0.52       299
```

```
precision    recall  f1-score   support

        0      1.00      0.14      0.25        57
        1      0.41      0.67      0.51        73
        2      0.39      0.22      0.28        59
        3      0.43      0.72      0.54        60
        4      0.62      0.47      0.53        51

 accuracy                          0.46       300
```

```
precision    recall  f1-score   support

        0      0.93      0.31      0.47        45
        1      0.41      0.64      0.50        58
        2      0.38      0.18      0.25        82
        3      0.39      0.68      0.50        71
        4      0.56      0.42      0.48        43

 accuracy                          0.44       299
```

```
precision    recall  f1-score   support

        0      0.49      0.44      0.46        57
        1      0.35      0.36      0.35        73
        2      1.00      0.02      0.03        59
        3      0.32      0.90      0.48        60
        4      0.50      0.06      0.11        51

 accuracy                          0.36       300
```

```
precision    recall  f1-score   support

        0      0.61      0.87      0.72        45
        1      0.37      0.53      0.44        58
        2      0.56      0.29      0.38        82
        3      0.47      0.65      0.55        71
        4      0.82      0.21      0.33        43

 accuracy                          0.50       299
```

Figure 3: Results of the Blended Approach, and the Vanilla Approach.

of utilizing dependency parsers in language models. Simply put, improving the results on a specified subset of tasks or models would suffice to reject our null hypothesis.

4.1 Results and Discussion

Our results are summarized in Table 3, where the two modes are depicted across various training and testing sets. The general picture emerging from the analysis is that the Blended approach achieves an accuracy of 45%, outperforming the Vanilla-based approach by 3% in accuracy. In this regard, we can reject our null hypothesis, meaning that additional fine-tuning with scenes built from dependency parsers affects the existing model performance.

Given that the number of training materials is limited, achieving 3% in accuracy with a significantly smaller standard deviation (0.05 vs. 0.13) shows that our blended results are closer to the mean, meaning they are not spread out. Moreover, the Blended approach's accuracy, recall, and F1 score are significantly better than the Vanilla approach (see Figure 3). Also, compared to the Vanilla approach, the blended mechanism offers stable results, which is very important for any NLP task.

Further research revealed that, on average, for 1300 reviews, 1389 triples were utilized to develop sentences —304 for the testing set, showing that the Scene Constructor played an important role in enhancing each training and testing sample.

Although there are probably better solutions, we consider this blended-triple-sentence mechanism advantageous because it automatically enhances each training sample on the fly for any given task. In this regard, the Blended approach can help the research community produce more accurate results with language models —this would further address problems when large amounts of training data are unavailable.

Finally, we want to point out that our results must be taken with a grain of salt as our methodology depends on the number and the quality of the training samples. In this regard, the Blended mechanism does not purport to replace other augmented solutions that the AI community is currently utilizing but only to help them as an additional strategy in their toolbox.

Table 3: Results of the Vanilla and the Blended approach.

Results		
-	Vanilla	Blended
Round	Accuracy	Accuracy
1st	0.47	0.37
2rd	0.21	0.46
3rd	0.46	0.41
4th	0.45	0.46
5th	0.43	0.36
6th	0.48	0.47
7th	0.17	0.46
8th	0.58	0.52
9th	0.49	0.44
10th	0.48	0.50
Average	0.42	0.45

5 CONCLUSION

This study used a simplified technique to analyze the relationship between dependency parsers and language models as a step toward understanding the impact of fine-tuning the latter with enhanced samples designed upon word relations found by the former. Given the successes of utilizing dependency parsers in developing classic AI systems, we suspected that further fine-tuning language models with semantics found in parsed samples would improve their accuracy results, especially when we do not have access to a large amount of training material.

Undertaken experiments revealed that enhancing language models with semantic relations improves their accuracy scores, indicating the robustness of this method. The success highlights the importance of utilizing dependency parsers when fine-tuning language models, suggesting that they seem to help the models generalize well when dealing with unseen classes or sequences of texts that were not met before.

We hope our work will spur further research on the relationship between parsers and language models. Note that we did not compare possible differences in utilizing the syntactic output of parsers. As this seems to be a rapidly evolving area of research, we would encourage researchers to build upon this work by examining the impact of utilizing and comparing results based on the syntactic versus semantic analysis of training samples.

REFERENCES

Bengio, Y. (2008). Neural net language models. *Scholarpedia*, 3(1):3881.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

He, P., Liu, X., Gao, J., and Chen, W. (2020). Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Isaak, N. and Michael, L. (2016). Tackling the Winograd Schema Challenge Through Machine Logical Inferences. In Pearce, D. and Pinto, H. S., editors, *STAIRS*, volume 284 of *Frontiers in Artificial Intelligence and Applications*, pages 75–86. IOS Press.

Isaak, N. and Michael, L. (2017). How the Availability of Training Material Affects Performance in the Winograd Schema Challenge. In *Proceedings of the (IJCAI 2017) 3rd Workshop on Cognitive Knowledge Acquisition and Applications (Cognitum 2017)*.

Isaak, N. and Michael, L. (2021). Experience and prediction: a metric of hardness for a novel litmus test. *Journal of Logic and Computation*, 31(8):2028–2056.

Kübler, S., McDonald, R., and Nivre, J. (2009). Dependency parsing. *Synthesis lectures on human language technologies*, 1(1):1–127.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Levesque, H. J. (2014). On Our Best behaviour. *Artificial Intelligence*, 212:27–35.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

McCarthy, J., Minsky, M. L., Rochester, N., and Shannon, C. E. (2006). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955. *AI magazine*, 27(4):12–12.

Michael, L. (2013). Machines with websense. In *Proc. of 11th International Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense'13)*.

Nivre, J. (2005). Dependency grammar and dependency parsing. *MSI report*, 5133(1959):1–32.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Rahman, A. and Ng, V. (2012). Resolving Complex Cases of Definite Pronouns: The Winograd Schema Challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 777–789, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rasooli, M. S. and Tetreault, J. (2015). Yara parser: A fast and accurate dependency parser. *arXiv preprint arXiv:1503.06733*.

Romera-Paredes, B. and Torr, P. (2015). An embarrassingly simple approach to zero-shot learning. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2152–2161, Lille, France. PMLR.

Sharma, A., Vo, N. H., Aditya, S., and Baral, C. (2015). Towards Addressing the Winograd Schema Challenge - Building and Using a Semantic Parser and a Knowledge Hunting Module. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, pages 25–31.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, C., Liu, X., and Song, D. (2020). Language models are open knowledge graphs. *arXiv preprint arXiv:2010.11967*.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2021). Fine-tuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Xian, Y., Schiele, B., and Akata, Z. (2017). Zero-shot learning-the good, the bad and the ugly. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4582–4591.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.