PARSE2VEC: TRANSFORMER EMBEDDINGS DEEPER THAN SURFACE FORMS

MICHAEL CONNOR

Abstract. **Objective:** The minimum transformer performance can move forward by incorporating the progress of the last decade. **Background:** Current efforts begin with surface forms and build meanings for tagging, parsing, identifying semantic relationships, etc. But a parser and outside resources can also provide these morphological, syntactic, and semantic features. **Method:** Instead of inputting tokens of surface forms into a transformer, we could start with token vectors whose summed embeddings represent the deep features of the words. This is an empirical study on how inputs with token vectors perform on masked word prediction.

Background

The latent information in a transformer model and in word embeddings covers a diverse field over more than a decade. The impetus for this work is (15), (13), and (14), who cataloged syntactic and semantic word relations, and showed how words are composable through their relationships. In (3), surface form words, root words, suffixes, and parts of speech form various feature vectors for word embedding reasoning tasks. In (11), surface forms and dependencies from spaCy combine as complex values for translations. In (6), spaCy output including sub-word tokens and dependencies create a gloss representation for translations. In (24), they showcased the internal model's ability to perform parsing related tasks while trained for masked word prediction. In (9) and (2), sub-word segments combine from Byte Pair Encoding plus a morphological segmentation table, in order to produce better sub-word tokenization. In (4), gendering in words introduces bias in word embeddings. In (20), entities and coreferences combine within the transformer during training. In (7), NER marking boosts the identification of legal clauses and metadata. In (21), among other findings, related absolute positions were found to be equivalent but preferred to learned absolute positions. In (18), relative positions outperform absolute positions in some tests. And the Transformer of (21) uses surface forms, but can easily accommodate multimodal input.

All of these seem unrelated, but all of these involve latent information within the transformer model and a parser can output facsimiles of all of these and attach outside resources[1]. Given the conclusions of past work that transformer models contain these relationships from surface form text, a transformer model with these relationships as input should have comparable results. Various multimodal fusion strategies exist from (21), (22), (1), and (19), which could fit this novel multimodal arrangement.

Method

The training corpus goes through the spaCy (8) parser and combines with outside csv tables. Among other parsers tested, spaCy has a free and publicly available version with ample documentation.

[1]https://github.com/maconnor34/parse2vec

Each input word is represented by a vector of features rather than by the surface form alone. A transformer using surface form input and with surface form output for masked word prediction marks the baseline for comparison. The tests cover the gradual addition of features in groups, with an individual focus on Stanford Dependencies. The transformer model is multimodal and uses early fusion. The baseline uses fusion by summation as in (21), which is compared to other common fusion functions.

Masked word prediction relates directly to the model training and reflects a basic function of a transformer model. The metrics for the experiment are accuracy and perplexity of predicted words.

Configuration The scale of this experiment fits within the constraints of the BabyLM challenge (5). The volume of data in training is 1.25M words with 13K unique words and in testing it is 250K words. This data comes from the AG News corpus (23) for training and testing. The experiments run on a MSI GF75 Thin laptop with a Nvidia RTX 3090. The transformer architecture settings are: sequence length = 16, attention heads = 4, layers = 3, hidden dimension = 256, feed forward dimension = 1024, feed forward Dropout = 0.1, embedding Dropout = 0.5, embedding LayerNorm epsilon = 1e-12, encoder LayerNorm epsilon = 1e-5, optimizer = Adam with no other settings.

Preprocessing: SpaCy produces *token* objects containing *text*, *tag_* which are Penn tags (17), *pos_* which are Universal Dependency Tags (16), *lemma_*, *dep_* which are Stanford tags (12), *morph*, and *ent_type_* from en_core_web_lg. A second parser instance runs to produce *coref_clusters_* tags from en_coreference_web_trf. Based on these, head dependency tags and more position tokens are created. Finally, outside csv tables are loaded to add root words, suffixes, sub-suffixes, and gender features.

Features like the elements of spaCy's *token* appear multiple times in previous work, often in pairings like the surface form and dependency couples of (6) and (11). Relating to dependencies, a head dependency tag is a copy of the dependency tag attached to the head of the compliment or modifier and marked as head. For example in "brown fox", *amod* attaches to "brown" and *head_amod* attaches to "fox". The data from *ent_type_* is included for its semantic categories as in (7), and also for its syntactic fit in the sentence. Incorporating coreferences happens here before the transformer and copies the practice of (20) in replacing the pronouns with the entities. Root words, suffixes and sub-suffixes are not sub-word tokens and are not defined by Byte Pair Encoding like in (9). They are copied from a morphological segmentation table with major suffix combinations. Gender features come from a list of common gendered nouns in order to flag obvious gendering but not debias it as in (4). There is no debiasing or normalizing here as all features have random initialization and are learned features. While there are no corrections, a greater number of explicit features leave less space for hidden relationships.

Multiple position features explicitly represent the complex relationships of word order. There are four positioning dimensions present: learned absolute positions, relative positions, relative positions within sets, and parse tree position. The absolute position token moves from its own set of tokens into the main feature set. Then there are relative positions like (18). Combining parser output with positions produces relative positions within sets of parts of speech such as $noun{-}2, noun{-}1, noun{+}1, noun{+}2$ to indicate the previous or subsequent nouns. In addition to their qualitative meanings, the dependency tags and their head dependency tags mark relative positions within parse trees.

The token vectors represent context words as the aggregate of their features as in (3). These vectors of feature tokens have exclusive subsets, $NONE$ features, and a hard vector length.

All the features of these token vectors require 3 times the original set of tokens for surface form features. The number of tokens for root words and lemmas scale with the number of surface form tokens. There are less than 1000 tokens for all other features. Masked words have only the $MASK$ feature and absolute position. Target words are restricted to surface_form tokens. The dimensions of the input far exceed that of the output.

There is a 5 word minimum frequency threshold, below which words are replaced with UNK. All digits are converted to NUM. Numbers that appear spelled out remain. All punctuation is dropped after spaCy uses it for tokenization. All embeddings for all features start with Xavier initialization.

For token vectors, minimum thresholding eliminates features but not entire words. Features of entries are not removed but replaced with UNK. Since other features provide a partial view, these are comparable to other incomplete entries like numbers whose numeric values are replaced with NUM.

Creating a view of the parser output that masks the target word requires multiple parser passes. A gibberish word replaces every 12th word to simulate it being unknown, and therefore 12 sets of modification annotations attach to every token. When marshalling the data for processing, the token vectors of every context word loose features as if the masked word was the 12th unknown word. Dependencies disappeared most commonly and at the greatest distance. The extra processing amounted to less than that of the coreference parsing pass in spaCy.

Processing: The classifier is a classic transformer encoder with source code from (10), a straight forward implementation of (21). A 32 dimension input replaces the 1 dimensional surface form with a 1 dimensional position. The fusion function occurs in the call to Embedding.

$$x = e_{surface\_form} + e_{position}$$
$$\mathrm{Q} = xA + b$$

The calculation of Q, K, and V remain the same, where each is a projection of the embeddings.

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$
$$x = x + \mathrm{Attention}(Q, K, V)$$

This is the classic method for passing the input into the transformer model. In (21), the $surface\_form$ word tokens and *position* tokens rest in two separate Embedding objects with different initializations. The embedding passes to 3 projection matrixes to form Q, K, and V. Finally the softmax function activates the dot product of Q and K, and then projects back into V to get an attention vector to add to the embedding vector.

$$x = \sum_{i=1}^{d_{word}} e_i$$

This calculation of x extends (21), the sum of embeddings of the dimensions of *word*, including the $surface\_form$ and *position*, and all features initialize together. The multimodal features come from an external source, a parser, but they stay within the internal meaning of the word, which the embedding presents to the model. This changes the lookup process for an embedding, and this changes the flow of learning within the model. The features do not hide the internal dimensions of single word representations, but rather name them and demarcate

them. Before, words that were nouns each had an unnamed noun dimension related to each noun and those dimensions had common unnamed noun dimensions. With token vectors, noun words have unnamed noun dimensions that relate to each word, but their common noun dimension has a feature name. In the classic transformer of (21), the embeddings of surface_form tokens begin as words and then relationships flow backward from predictions into feed forward networks into attention heads of the model and into the word embeddings during training. Word token vectors provide words and relationships, so that these internal representations flow in both directions.

Other established multimodal early fusion algorithms are tested as well.

$$x = norm(W)\mathbf{e}$$

This calculation of x uses a vector of normalized weights W of dimension 1 X d_{word} to produce a weighted sum by matrix multiplication, as in (22).

$$g = Concat(e_1, ..., e_d)$$
$$x = gW$$

This calculation of x uses a concatenation of the embeddings of e and a projection matrix W of dimension $d_{word} * embedding\ dimension\ X\ embedding\ dimension$, also described in (22).

$$\mathbf{h} = tanh(\mathbf{e}W)$$
$$k = Concat(h_1, ..., h_d)$$
$$\mathbf{z} = sigmoid(Vk)$$
$$x = \sum \mathbf{hz}$$

This calculation of x uses a Gated Multimodal Unit from (1). W and V are learned weights vectors of dimensions $embedding\ dimension\ X\ d_{word}$ and $d_{word} * d_{word}\ X\ d_{word}$.

$$\mathbf{h} = tanh(\mathbf{e}W)$$
$$k = \sum_{i=1}^{d_{word}} h_i$$
$$\mathbf{z} = sigmoid(Vk)$$
$$x = \mathbf{zh}$$

This calculation of x uses a gated fusion function based on a sum instead of concatenation. W and V are learned vectors of weights of dimension $embedding\ dimension\ X\ embedding\ dimension$.

$$x = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

This calculation of x uses the self attention formula in order to reduce the multimodal dimensions similar to its use in (19).

$$a = e_{surface\_form}$$
$$b = e \in [1, ..., d-1]\ all\ dimensions\ d\ except\ surface\_form$$
$$x = \text{softmax}\left(\frac{Q_a K_b^T}{\sqrt{d}}\right)V_b$$

TABLE 1. Token vs Token Vectors of Various Features

Token Vector Size	Features	Accuracy	Perplexity
2	Surface Form + Absolute Position	20.43	1314.20
22	+ pos_ + Root + Suffixes + tag_ + lemma_ + ent_type_ + morph + Sub-Suffixes	24.83	624.44
23	+ Dependency (dep_)	27.05	569.70
26	+ Head Dependency	32.31	320.26
32	+ Gender + Coreference + Relative Positions	33.47	361.33

This calculation of x uses the cross attention in order to reduce the multimodal dimensions, also covered in (19).

Experiments

In all training, every other epoch writes a model until 40 training epochs. The results below are the maximum accuracy as a percent and its accompanying perplexity. Nearly all models encountered overfitting by 40 epochs. The baseline began overfitting at epoch 10.

There are 5 tests in Table 1, beginning with the baseline and then adding features.

Adding the parser features improves accuracy by 63% and reduces perplexity by 73%. Drops in perplexity correlate with the rises in accuracy. The added features appear in groups to show the progression of improvements. The greatest contributions come from the features for dependencies and dependency heads. Aside from dependency features, no single feature could overcome the noise of random initializations. Therefore further ablative testing was abandoned, and they are bundled for testing.

These results show an aggregation of performance increases comparable with those in previous works concerning word prediction. Cross Entropy Loss improvements were 5-15% in (2) for next word prediction. Accuracy improvements were 1.7-9.7% in (3) for sentence completion (predict the last word in a sentence), 6% in (14) for sentence completion, 13-25% in (9) for last word prediction (predict the last word in a paragraph). Other experiments use BLEU scores and other classification or reasoning tasks, with most improvements in the range of 10-25%.

Other classic measurements of improvement become moot or lose impact with token vectors. The fertility of sub-word tokens does not increase from 1 when they combine in a token vector. Fertility does not vary or respond to learning as in (9) and (2), and sub-word tokens do not need to account for statistical efficiency with sequence lengths. With word cosine similarity, the overlap of composable token vectors overshadows their difference. Word relationships now carry features that appear on both sides. For a Word2Vec (13) example of X_{apple} - X_{apples} $\approx$ X_{car} - X_{cars}, the left side expression X_{apple} - X_{apples} could be represented by [$apple_{string}$, $apple_{lemma}$, $apple_{root}$, $noun$, $singular$, $Suffix_{NONE}$] - [$apples_{string}$, $apple_{lemma}$, $apple_{root}$, $noun$, $plural$, $Suffix_s$]. There is a difference between surface forms, but more importantly a difference of $plural$ + $Suffix_s$ and $singular$ + $Suffix_{NONE}$, as well as a similarity of $apple_{lemma} + apple_{root} + noun$.

There are 7 early fusion functions tested in Table 2. All early fusion experiments use the full 32 feature input.

Summation had the highest single accuracy measurement, but the next 3 belonged to Weight Vector, the closest function. The increasing complexity of the functions correlates to their decreasing performance. However, the lesser performances of the other fusion functions have no clear explanation and might change under different circumstances. Since all features are learned and all relate to the words, some multimodal fusion functions may not have anything

TABLE 2. Early Fusion Functions

Fusion Function	Accuracy	Perplexity
Summation	33.47	361.33
Weight Vector	33.27	488.98
Concatenation	31.81	402.97
Gated Multimodal Unit	30.71	428.40
Gated Fusion	31.88	495.29
Self Attention	30.39	485.73
Cross Attention	32.89	357.09

truly different to fuse. The Gated Fusion function based on summation performs better on accuracy than the classic Gated Multimodal Unit based on concatenation, similarly to the Summation and Concatenation functions. This may also reflect that all the features fit together as different views of words.

Conclusions

Additional inputs start a model on a greater foundation of semantic content and of syntactic form for representing relationships of language. A transformer model based on the surface forms might eventually contain similar implicit syntactic and semantic relationships as latent relationships from the data set. Explicitly moving a large segment of this information into the input raises the floor of model performance and generalization as measured by masked word prediction accuracy and perplexity. The extra dimensions of the multimodal input do not interact multimodally and might be better described as the deeper meaning of the words.

References

(1) J. Arevalo et al. *Gated Multimodal Units for Information Fusion*. 2017. arXiv: 1702.01992 (stat.ML). URL: https://arxiv.org/abs/1702.01992.

(2) E. Asgari, Y. El Kheir, and M. Javaheri. "MorphBPE: A Morpho-Aware Tokenizer Bridging Linguistic Complexity for Efficient LLM Training Across Morphologies". In: (Feb. 2025). DOI: 10.48550/arXiv.2502.00894.

(3) J. Bian, B. Gao, and T.-Y. Liu. "Knowledge-Powered Deep Learning for Word Embedding". In: *Machine Learning and Knowledge Discovery in Databases*. Nancy France: Springer-Verlag, 2022, pp. 132–148. ISBN: 978-3-662-44847-2. DOI: 10.1007/978-3-662-44848-9_9. URL: https://doi.org/10.1007/978-3-662-44848-9_9.

(4) T. Bolukbasi et al. *Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings*. 2016. arXiv: 1607.06520 (cs.CL). URL: https://arxiv.org/abs/1607.06520.

(5) L. Choshen et al. *BabyLM Turns 4: Call for Papers for the 2026 BabyLM Workshop*. 2026. arXiv: 2602.20092 (cs.CL). URL: https://arxiv.org/abs/2602.20092.

(6) S. Egea Gómez, E. McGill, and H. Saggion. "Syntax-aware Transformers for Neural Machine Translation: The Case of Text to Sign Gloss Translation". In: *Proceedings of the 14th Workshop on Building and Using Comparable Corpora (BUCC 2021)*. Ed. by R. Rapp, S. Sharoff, and P. Zweigenbaum. Online (Virtual Mode): INCOMA Ltd., Sept. 2021, pp. 18–27. URL: https://aclanthology.org/2021.bucc-1.4/.

(7) C. Han and S. Jalagam. *Metadata Extraction Leveraging Large Language Models*. 2025. arXiv: 2510.19334 (stat.ML). URL: https://arxiv.org/abs/2510.19334.

(8) M. Honnibal et al. *spaCy: Industrial-strength Natural Language Processing in Python*. 2020. DOI: 10.5281/zenodo.1212303. URL: https://doi.org/10.5281/zenodo.1212303.

(9) H. Jabbar. *MorphPiece : A Linguistic Tokenizer for Large Language Models*. 2024. arXiv: 2307.07262 (cs.CL). URL: https://arxiv.org/abs/2307.07262.

(10) L. v. W. Lewis Tunstall and T. Wolf. *Natural Language Processing with Transformers*. O'Reilly, 2022. ISBN: 978-1-098-13679-6.

(11) Y. Liu and Y. Hou. *Syntax-Aware Complex-Valued Neural Machine Translation*. 2024. arXiv: 2307.08586 (cs.CL). URL: https://arxiv.org/abs/2307.08586.

(12) M.-C. de Marneffe and C. D. Manning. "The Stanford Typed Dependencies Representation". In: *Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*. Manchester, UK, 2008, pp. 1–8. URL: https://aclanthology.org/W08-1301.pdf.

(13) T. Mikolov, W.-t. Yih, and G. Zweig. "Linguistic Regularities in Continuous Space Word Representations". In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by L. Vanderwende, H. Daumé III, and K. Kirchhoff. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 746–751. URL: https://aclanthology.org/N13-1090/.

(14) T. Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of Workshop at ICLR* 2013 (Jan. 2013).

(15) T. Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *CoRR* abs/1310.4546 (2013). arXiv: 1310.4546. URL: http://arxiv.org/abs/1310.4546.

(16) J. Nivre et al. "Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection". In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 4034–4043. URL: https://aclanthology.org.

(17) B. Santorini. *Part-of-speech tagging guidelines for the Penn Treebank Project*. Tech. rep. MS-CIS-90-47. Philadelphia, PA, USA: Department of Computer and Information Science, University of Pennsylvania, 1990. URL: ftp://ftp.cis.upenn.edu/pub/treebank/doc/tagguide.ps.gz.

(18) P. Shaw, J. Uszkoreit, and A. Vaswani. *Self-Attention with Relative Position Representations*. 2018. arXiv: 1803.02155 (cs.CL). URL: https://arxiv.org/abs/1803.02155.

(19) Z. Shen et al. *Rethinking Early-Fusion Strategies for Improved Multimodal Image Segmentation*. 2025. arXiv: 2501.10958 (cs.CV). URL: https://arxiv.org/abs/2501.10958.

(20) N. Stylianou and I. Vlahavas. "E.T.: Entity-Transformers. Coreference augmented Neural Language Model for richer mention representations via Entity-Transformer blocks". In: *Proceedings of the Third Workshop on Computational Models of Reference, Anaphora and Coreference*. Ed. by M. Ogrodniczuk et al. Barcelona, Spain (online): Association for Computational Linguistics, Dec. 2020, pp. 1–10. URL: https://aclanthology.org/2020.crac-1.1/.

(21) A. Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

(22) X. Wang et al. "Multi-Modal Generative AI: Multi-Modal LLMs, Diffusions, and the Unification". In: *IEEE Transactions on Circuits and Systems for Video Technology* 36.4

(Apr. 2026), pp. 5621–5641. ISSN: 1558-2205. DOI: 10.1109/tcsvt.2025.3635224. URL: http://dx.doi.org/10.1109/TCSVT.2025.3635224.

(23) X. Zhang, J. Zhao, and Y. LeCun. "Character-level Convolutional Networks for Text Classification". In: *Advances in Neural Information Processing Systems 28*. 2015, pp. 649–657. URL: http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html.

(24) H. Zhao et al. "Do Transformers Parse while Predicting the Masked Word?" In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by H. Bouamor, J. Pino, and K. Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 16513–16542. DOI: 10.18653/v1/2023.emnlp-main.1029. URL: https://aclanthology.org/2023.emnlp-main.1029/.

Email address: maconnor34@gmail.com