TRACKING THE PROGRESS OF LANGUAGE MODELS BY EXTRACTING THEIR UNDERLYING KNOWLEDGE GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

The state of the art, previously dominated by pre-trained word embeddings, is now being pushed forward by large pre-trained contextual representation models. This success has driven growing interest to understand what these models encode inside their inner workings. Despite this, understanding their semantic skills has been elusive, often leading to unsuccessful, non-conclusive, or contradictory results among different works. In this work, we define a probing classifier that we then use to extract the underlying knowledge graph of nine of the most influential models of the last years, including word embeddings, context encoders, and text generators. This probe is based on concept relatedness, grounded on WordNet. Our results show that this knowledge is present in all the models, but has several inaccuracies. Furthermore, we show that the different pre-training strategies and architectures lead to different model biases. We conduct a systematic evaluation to discover specific factors that explain why some concepts are challenging for the different families of models. We hope our insights will motivate the future development of models that capture concepts more precisely.

1 INTRODUCTION

Natural language processing (NLP) encompasses a wide variety of applications such as summarization (Kovaleva et al., 2019), information retrieval (Zhan et al., 2020), and machine translation (Tang et al., 2018), among others. Currently, the use of pre-trained language models has become the *de facto* starting point to tackle most of these applications. The usual pipeline consists of finetuning a pre-trained language model by using a discriminative learning objective to adapt the model to the requirements of each specific task. As a key ingredient, these models are pre-trained using massive amounts of unlabeled data that can include millions of documents. Furthermore, current models have a large size that can include billions of parameters. Massive data and parameters are supplemented with a suitable learning architecture, resulting in a highly powerful but also complex model, whose internal operation is hard to analyze.

The success of pre-trained language model has driven the interest to understand the mechanisms that they use to solve NLP tasks. As an example, in the case of BERT (Devlin et al., 2019), one of the most popular pre-trained models based on a Transformer architecture (Vaswani et al., 2017), several studies have attempted to access the knowledge encoded in its layers and attention heads (Tenney et al., 2019b; Devlin et al., 2019; Hewitt & Manning, 2019). In particular, (Jawahar et al., 2019) show that BERT can solve tasks at a syntactic level by using Transformer blocks to encode a soft hierarchy of features at different levels of abstraction. Similarly, (Hewitt & Manning, 2019) show that BERT is capable of encoding structural information from text. In particular, using a structural probe, they show that syntax trees are embedded in a linear transformation of the encodings provided by BERT.

In general, previous efforts have provided strong evidence indicating that current pre-trained language models encode complex syntactic rules, however, relevant evidence about their abilities to capture semantic information remains still elusive. As an example, a recent study (Si et al., 2019) attempts to locate the encoding of semantic information as part of the top layers of Transformer architectures, however, results provide contradictory evidence. Similarly, (Kovaleva et al., 2019)

focus on studying knowledge encoded by self-attention weights, however, results provide evidence for over-parameterization but not about language understanding capabilities.

In this work, we address the challenging question about to what extent pre-trained language models encode semantic information. As a key source of semantic knowledge, we focus on studying to what extend pre-trained language models can encode the semantic embedded in the conceptual taxonomy of WordNet (Miller, 1995). The ability to understand, organize, and correctly use concepts is one of the most remarkable capacities of human intelligence (Lake et al., 2017). Therefore, a quantification of the ability that a pre-trained language model can exhibit to encode the conceptual organization behind WordNet is highly valuable. In particular, it can provide useful insights about the inner mechanisms that these models use to encode semantic information. Furthermore, an analysis of concepts and associations that result difficult to these models can provide relevant insights about how to improve them.

In contrast to most previous works, we do not focus on a particular model, but we embrace our research in general terms by considering a large list of the most popular pre-trained language and text-embedding models. In this sense, one of our goals is to provide a comparative analysis of the capacities of different types of approaches. Following Hewitt & Manning (2019), we study semantic performance by defining a probing classifier based on concept relatedness according to WordNet. Using this tool, we analyze the different models, enlightening how and where semantic knowledge is encoded. Furthermore, we analyze how these models encode suitable information to recreate the structure of WordNet. Among our main results, we show that the different pre-training strategies and architectures lead to different model biases. In particular, we show that contextualized word embeddings, such as BERT, encode high-level concepts and hierarchical relationships among them, creating a taxonomy. This finding corroborates results presented in previous work (Reif et al., 2019) that claim that BERT vectors are stored in linear sub-spaces that have correspondence with semantic knowledge. On the other hand, our results show that all the analyzed models have a high correlation with each other in terms of the concepts that are hard or easy to capture. Our study also shows evidence about the limitations of current pre-trained language models, demonstrating that all of them have difficulties to encode specific concepts. As an example, across all the models of the family of BERT, performance degrades as the predicted concept or relation is closer to the leaves of the WordNet taxonomy. Our results also reveal that models have very distinctive patterns in terms of where they encode most of the semantic information. These patterns are dependant on architecture and not on model sizes.

This work is organized as follows. Section 3 introduces the methodology used to conduct this study, introducing the probing classifier, and specifying the method used to reconstruct the WordNet taxonomy. Section 4 shows the results obtained using the probing classifier, discussing how accurate the semantic knowledge of each model is and showing the results of the reconstruction of WordNet. Section 5, uncovers which instances of our tests are more difficult to solve for each of these models, revealing their main limitations. Section 6 reveals which blocks of these architectures have a correspondence to the semantic tasks of the study. Finally, in Section 7 we discuss the implications of our findings along with concluding remarks.

2 Related work

The success of deep learning architectures in various NLP tasks has fueled a growing interest from the community to improve understanding of what these models encode. Several works have studied these models' impact on downstream tasks at the syntactic (Goldberg, 2019; Hewitt & Manning, 2019) or semantic level (Ettinger, 2020). Some studies (Tenney et al., 2019b) claim that success in a specific task helps understand what type of information the model encodes. Other studies have improved the understanding of how and why these models work by analyzing correlations between input-targets and specific architecture blocks, such as layers (Jawahar et al., 2019), encoded hidden states (Tang et al., 2018), and attention heads (Michel et al., 2019).

Using probing classifiers, Clark et al. (2019) claims that some specific BERT's attention heads show correspondence with syntactic tasks. They showed that some heads on the same layer have similar behavior. Goldberg (2019) illustrates the capabilities that BERT has to solve syntactic tasks, such as subject-verb agreement. BERT's success in these tasks fuels the belief that BERT can code the syntax of a language. Hewitt & Manning (2019) propose a structural probe that evaluates whether syntax

trees are embedded in BERT embeddings' linear transformation. The study shows that such transformation exists in BERT, providing evidence that syntax trees are implicitly embedded in BERT's vector geometry. Reif et al. (2019) have found evidence of syntactic representation in BERT's attention matrices, with certain directions in space representing particular dependency relations. The study shows that just as there are specific syntactic subspaces, there is evidence for subspaces representing semantic information. The authors suggest that BERT's internal geometry may be broken into multiple linear subspaces, with separate spaces for different syntactic and semantic information.

The ability of Transformer-based models to solve tasks at the semantic level is more debatable. While some studies show satisfactory results in tasks such as entity types (Tenney et al., 2019a), semantic roles (Rogers et al., 2020), and sentence completion (Ettinger, 2020), other studies show less favorable results in coreference (Tenney et al., 2019b) and Multiple-Choice Reading Comprehension (Si et al., 2019), claiming that BERT's performance may not reflect the model's true ability of language understanding and reasoning. Some works have studied which blocks of BERT are used to solve tasks at the semantic level. Tenney et al. (2019b) proposed a set of edge probing tasks to test the encoded sentential structure of contextualized word embeddings. The study shows evidence that the improvements that BERT and GPT-2 offer over non contextualized embeddings as GloVe is only significant in syntactic-level tasks. Tenney et al. (2019a) combine information from the many layers of a BERT encoder to solve probing tasks. The study shows that the first layers of the encoder better solve tasks that require short dependency relationships at the syntactic level (e.g., subject-verb agreement) and that tasks that require long-range dependencies (e.g., subject-object dependencies) require information encoded in the top layers. While these studies also claim that the top layers appear to encode semantic information, the evidence to support this claim is not conclusive. For example, Jawahar et al. (2019) only manages to locate one SentEval semantic task topped at the last layer.

Some studies have focused on understanding how BERT layers encode information. Voita et al. (2019a) found that information about the past in left-to-right language models gets vanished as the information flows from bottom to top BERT's layers. Press et al. (2020) evaluated different patterns to combine the BERT layers, finding that more self-attention at the bottom and more feed-forward sub-layers at the top produced better results in language model tasks. Hao et al. (2019) showed that the lower layers of BERT are more invariant during finetuning, suggesting that the layers close to input learn more transferable language representations. Other studies have focused on understanding and visualizing how helpful self-attention heads are to solve some specific tasks (Vig, 2019). Kovaleva et al. (2019) study attention weights based on results obtained from tasks defined in GLUE (Wang et al., 2018), showing a set of attention patterns that is repeated across different heads. Michel et al. (2019) show that several heads can be removed from trained Transformer models without degradation in machine translation tasks. This result coincides with those found by Voita et al. (2019b).

In summary, on the one hand, the related work shows results that provide strong evidence concluding that BERT and other models based on the Transformer architecture can encode information at the syntactic level. Furthermore, BERT uses this information to solve various benchmark NLP tasks. Working with ambiguous words has allowed BERT to perform well on MT tasks, outperforming other architectures in WSD. On the other hand, while some studies claim that top layers are helpful to solve semantic tasks, the results are not conclusive. Efforts to locate which blocks of the Transformer mechanisms operate at the semantic level have been unsuccessful. In this context, our work fills the gap between the better understanding that we have about word embeddings and language models capacities to work with syntax, and the shallow comprehension of their abilities to encode and work with semantics.

3 Study methodology

3.1 Word embeddings models

This study considers the most influential Language Models from recent years. We considered the essential approaches of three model families: non contextualized word embeddings (NCE), contextualized word embeddings (CE), and generative language models (GLM). We consider Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) for the first family of approaches. For the CE family, we consider ELMo (Peters et al., 2018), which is implemented on a bidirectional

LSTM architecture, XLNet (Yang et al., 2019), and BERT (Devlin et al., 2019) and its extensions ALBERT (Lan et al., 2020) and RoBERTa (Liu et al., 2019), all of them based on the Transformer architecture. GPT-2 (Radford et al., 2018) and T5 (Raffel et al., 2019) were included in the study to incorporate approaches based on generative language models.

3.2 SEMANTIC PROBING CLASSIFIER

We define an edge probing classifier that learns to identify if two words are semantically related. To create the probing classifier, we retrieve all the glosses from the Princeton WordNet Gloss Corpus[1]. This dataset provides WordNet's synsets gloss sentences with annotations identifying occurrences of concepts. The annotations provide a mapping of the used words to their corresponding WordNet node. We sample hypernym pairs A, B. We sample a third synset C, at random from an unrelated section of the taxonomy, taking care that C is not related to either A or B. Then, $\langle A, B, C \rangle$ form a triplet that allows us to create six testing edges for our classifier. To train the probing classifier, we define a labeled edge $\{x, y, L\}$, with x and y synsets in $\{A, B, C\}$, $x \neq y$. L is the target of the edge, L in $\{-1, +1\}$. If y is direct or indirect parent of x, $L = 1$, while $L = -1$ in other case. For each synset x, y, we sample one of its glosses $S(x)$, $S(y)$ from the dataset. Let M be a word embedding model. If M is a non contextualized word embedding model, x and y are encoded by $M(x)$ and $M(y)$, respectively. If M is a contextualized word embedding or a generative language model, x and y are encoded by the corresponding token of $M(S(x))$ and $M(S(y))$, respectively. Accordingly, the gloss of each word provides a context for the creation of its word embedding.

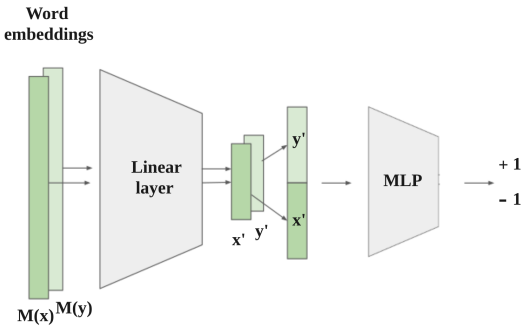


Figure 1: Inputs to the edge probing classifier correspond to the word embeddings M(x) and M(y) of words x and y, respectively. First, to account for the different dimensionality of the tested models, M(x) and M(y) are projected into a common lower dimensionality space using a linear layer. The resulting embeddings x' and y' are concatenated into a vector, which is fed into a MLP that is in charge of predicting if the word pair is related or not.

To accommodate for the evaluation of word embeddings of different dimensionalities, we first project the embeddings of each word x and y into a lower dimensionality space using a linear layer (see Fig 1). Then, these vectors denoted as x' and y', are concatenated and fed into a Multi-Layer Perceptron (MLP) classifier. The linear layer and the MLP are the only trainable parameters of our setting, as we use the source model weights without any finetuning.

Similar structural probes have been proposed to evaluate syntax encoding in BERT. However, we use an MLP to learn the structural relation between word pairs, providing the test with a mechanism that allows the embeddings to be combined in a non-linear way. Tests based on linear transformations such as that proposed by Hewitt & Manning (2019) did not allow us to recover the WordNet structure, which indicates that the subspaces in which the word embeddings models encode the semantics are not linear.

3.3 RECONSTRUCTING THE STRUCTURE OF THE KNOWLEDGE GRAPHS

Given a graph G with nodes N and unknown edges E, we define an auxiliary graph G' with nodes N and edges E', comprised of all possible directed edges. For each edge $e \in E'$, we obtain a

[1] https://wordnetcode.princeton.edu/glosstag.shtml

prediction h_e that estimates the probability of that edge representing a valid hypernymy relation, and a distance d_e that estimates the hypernym distance [2] between the nodes in G.

We define $\delta(v)$ to be the set of edges $\{\langle u, v\rangle : u \in N, u \neq v\}$ where the edge $\langle u, v\rangle$ defines a $\langle \text{parent}, \text{child}\rangle$ relation. We also define $\gamma(S)$ as the set of edges $\{\langle u, v\rangle \in E' : u \notin S, v \in S\}$. We estimate the graph topology of G defined by $E \subset E'$, by solving this optimization problem:

$$\max_{r \in N} \quad \sum_{e \in E'} x_e h_e \tag{1}$$

$$\text{s.t.} \quad x_e \in \arg\min \left[\sum_{e \in E'} x_e d_e \qquad \text{s.t.} \begin{cases} x_e \in \{0,1\} & e \in E' \\ \sum_{e \in \delta(v)} x_e = 1 & \forall v \in N \setminus \{r\} \\ \sum_{e \in \gamma(S)} x_e \geq 1 & \forall S \subset N \setminus \{r\} \end{cases} \right] \tag{2}$$

where the objective function (1) is used to find the best root node r and the nested optimization problem (2) is the minimum spanning arborescence problem applied to the dense graph G'. The final binary values of x_e estimate E by indicating if every possible edge e exist in the graph or not.

To solve this optimization problem, we need estimates of h_e and d_e for each edge e. We use the probing classifier's output probability as an estimate of h_e, and propose the following two alternatives to estimate d_e.

Model Confidence Metric (MCM): As we will discuss in Section 5, all nine models in this study tend to capture close concept relations more precisely than distant links. In practice, that originates that a concept like *chihuahua* will be matched with its parent node *dog* with higher confidence than its parent node *entity*. Thus the score $1 - h_e$ is a reasonable metric for d_e.

Transitive Intersections Metric (TIM): As the before mentioned pattern is only a fortunate side effect that may not hold true for future models, we explored another metric grounded directly in the tree structure of the underlying knowledge graph. The key observations are that nodes *u* and *v* that form a parent-child relation have some transitive connections in common. Specifically, all descendants of *v* are also descendants of *u*, and all the ancestors of *u* are also ancestors of *v*. Furthermore, the closer the link between *u* and *v* in the graph, the bigger the intersection. With that in mind, for each candidate edge $e = \langle u, v\rangle$ we define the following alternative metric for d_e:

$$C - \left(\sum_{j \in N \setminus \{u,v\}} h_{\langle u,j\rangle} h_{\langle v,j\rangle} + \sum_{j \in N \setminus \{u,v\}} h_{\langle j,u\rangle} h_{\langle j,v\rangle} \right) * h_{\langle u,v\rangle} \tag{3}$$

The reason why we included $h_{\langle u,v\rangle}$ at the end is to consider the strength of the edge direction. Constant C was added to force d_e to be positive.

4 How accurate is their semantic knowledge?

4.1 Semantic probing classifier results

Table 1 shows the results obtained using the edge probing classifier in many variants of the models studied. Results show that regardless of model sizes, performance is homogeneous within each family of models. Additionally, results show that NCE and GLM methods obtain a worse performance than those achieved by CE methods. In particular, CE family's differences over NCE are significant, reaching around 5 points of improvement in F1. When using the base model, some models improve their performance while others decrease it, with ELMo being the only one with a considerable improvement in performance when using the large version of the model (3.7%).

Notice, though, that all the metrics in Table 1 are pair-wise metrics, not graph metrics. As we are dealing with graphs, predicted edges are built upon related predicted edges, thus drifts in small regions of the graph may cause large drifts in downstream connections. Furthermore, our setup balances positive (links) and negative samples (not links), but in a real reconstruction scenario, the proportion of negative samples will be considerably larger. As a consequence, we emphasize that these numbers must not be considered in isolation, but together with the results reported in sections 4.2 and 4.3.

[2] The value of this distance will be small if the hypernym relation is close, or large if it is distant or not valid.

Family	Model	Emb. Size	F1-score	Accuracy
NCE	W2V (Mikolov et al., 2013)	300	.7683 ± .0135	.7427 ± .0124
	GloVe-42B (Pennington et al., 2014)	300	.7877 ± .0084	.7617 ± .0078
GLM	GPT-2 (Radford et al., 2018)	9984	.7862 ± .0132	.7858 ± .0099
	T5-small (Raffel et al., 2019)	7168	.8156 ± .0098	.8075 ± .0078
	GPT2-xl (Radford et al., 2018)	78400	.7946 ± .0151	.7910 ± .0102
	T5-large (Raffel et al., 2019)	51200	.8148 ± .0119	.7988 ± .0113
CE	ELMo-small (Peters et al., 2018)	768	.7986 ± .0126	.7872 ± .0106
	BERT-base (Devlin et al., 2019)	9984	.8240 ± .0123	.8234 ± .0100
	RoBERTa-base (Liu et al., 2019)	9984	.8392 ± .0100	.8349 ± .0083
	XLNet-base (Yang et al., 2019)	9984	.8306 ± .0113	.8267 ± .0092
	ALBERT-base (Lan et al., 2020)	9984	.8184 ± .0222	.8145 ± .0143
	ELMo-large (Peters et al., 2018)	3072	.8311 ± .0090	.8242 ± .0076
	BERT-large (Devlin et al., 2019)	25600	.8178 ± .0152	.8221 ± .0115
	RoBERTa-large (Liu et al., 2019)	25600	.8219 ± .0159	.8258 ± .0121
	XLNet-large (Yang et al., 2019)	25600	.8211 ± .0142	.8215 ± .0105
	ALBERT-xxlarge (Lan et al., 2020)	53248	.8233 ± .0107	.8215 ± .0086

Table 1: Results obtained using the edge probing classifier. We study the performance in many model's variants, considering small and large versions of several models. Results are shown grouped by families of methods indicated by NCE (non contextualized embeddings), GLM (generative language models), and CE (contextualized embeddings).

4.2 Extracting the Underlying Knowledge Graph

Predicting a knowledge graph has a complexity of at least $O(N^2)$ in the number of analyzed concepts. In our case, this imposes a highly demanding computational complexity because WordNet has over 82000 noun synsets. To accelerate experimentation and facilitate our analysis and visualizations, we focus on extracting a WordNet sub-graph that comprises only 46 nodes. These nodes were picked to create a target graph with easily recognizable relations. We use the tree-edit-distance to evaluate how close are the reconstructed graphs to the target graph extracted from WordNet. Table 2 shows our results.

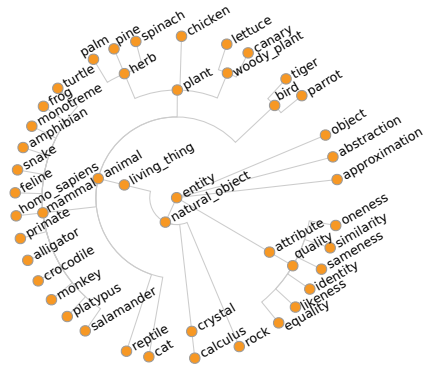


Figure 2: A reconstructed graph using MCM over BERT-large. The graphs for all the models can be found in Appendix B.

Family	Model	Tree Edition Dist.		
		TIM	MCM	Avg.
NCE	Word2Vec	59	59	**59**
	GloVe-42B	56	60	**58**
GLM	GPT-2	53	57	**55**
	T5	58	55	**56**
CE	ELMo	52	55	**53**
	BERT	49	48	**49**
	RoBERTa	56	54	**55**
	XLNet	52	48	**50**
	ALBERT	53	50	**51**

Table 2: Tree Edition Distance against the ground truth graph (large models used). We display both strategies for estimating d_e along with its average.

Table 2 shows that the knowledge graphs retrieved using the CE family of methods are closer to the target than the other graphs. In particular, the best results are achieved by BERT, ALBERT, and XLNet, showing that these models have better capabilities for encoding semantic information than their competitors. These results are consistent with the results obtained by the semantic probing classifier.

Visual inspection of the reconstructed graphs reveal that models capture some important categories, but fail to map some fine-grained relations.

4.3 ARE ALL THESE MODELS CORRELATED?

In this section, we study if concepts that are easy/hard to associate for one model are also easy/hard to associate for other models. We do this by analyzing the correlation between outputs using the semantic probing classifier. Graphs in Figure 3 show the correlation among the models by plotting the F1-score of one model against the rest. We picked one model of each family: GloVe (Fig 3-left) RoBERTa-large (Fig 3-middle), and T5-large (Fig 3-right).

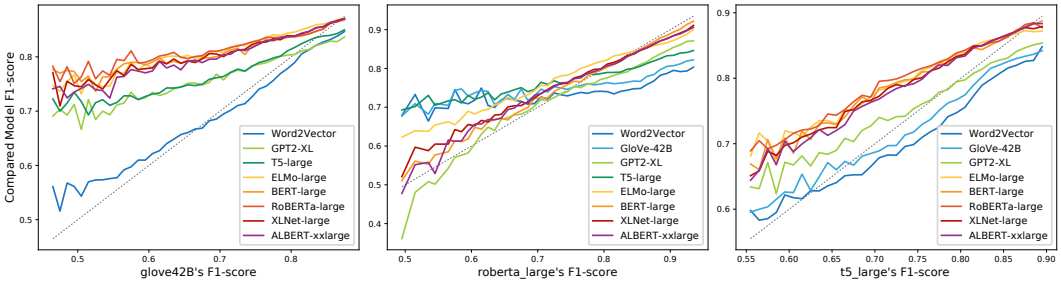


Figure 3: Analysis of F1-score correlations among models. A horizontal line in these graphs would represent perfect independence.

The graphs in Figure 3 reveal that correlations within each model family are strong, meaning that they tend to acquire similar biases. Figure 3-middle shows a strong correlation between RoBERTa and other methods in its family, but weaker with models of the NCE family. Fig 3-right shows that T5 has the same difficulties as the rest of the methods, even showing some correlation with the NCE family of methods.

5 WHAT IS EASY OR HARD? WHAT ARE THESE MODELS LEARNING?

The previous section shows that models from different families differ in their errors, but that models from the same families have similar biases. In this section, we elucidate which semantic factors have a higher impact on the performance of these models.

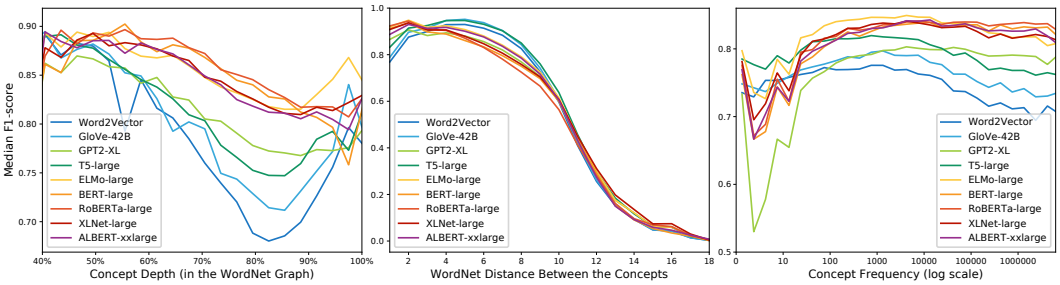


Figure 4: Semantic factors with a high impact on F1. Every plot displays only the range where at least 100 samples existed. From left to right, the graphs show concept depth, concept distance, and concept frequency. Several other factors are analyzed and visualized in Appendix C.

Relative depth in the WordNet graph: For each synset, we compared F1 with depth score (0 % for the root and 100 % for leaves) measuring differences between higher/lower level concepts. As seen in Figure 4-left, NCE-based methods are significantly impacted by this variable.

Graph distance between concepts: We measured the impact of the number of "hops" that separate two tested concepts on the pair-wise F1 score. Figure 4-middle reveals a strong correlation of all the models in this aspect. As an example of this phenomenon, closer relations such as $\langle$chihuahua, dog$\rangle$ are, in general, considerably easier to capture than distant relations such as $\langle$chihuahua, entity$\rangle$. For details on how we implement the distance in WordNet, check Appendix A.2.

Concept frequency: In Figure 4-right, we evaluate if frequent concepts are easier or harder to capture for these models. The frequency was computed by counting occurrences in the 38 GB of

OpenWebText Corpus[3]. The graph shows that uncommon and highly frequent concepts are harder to be modeled. In particular, NCE-based models and T5 are more sensitive to this factor. While the behavior related to frequent concepts seems to be counter-intuitive, we hypothesize that this corresponds to a phenomenon similar to what stop-words exhibit in language. Their unspecific use confused the models.

Number of child nodes: We discovered that all models have minimum F1-score for concepts with one or two child nodes (considering both direct and indirect children). After that point, they all increase their F1 as the number of child nodes increase. The same phenomenon shows up when we analyze only direct children. We hypothesize that concepts with a greater number of sub-categories appear in more diverse contexts in the models' original training corpus, thus models learn more accurate representations for them. The graph of this variable is in Appendix C.

6 WHERE IS THIS KNOWLEDGE LOCATED?

As mentioned in Section 2, prior work has not shown consensus on where is the semantic information located inside these architectures. Our experiments shed light on this subject. In Figure 5 we show how each layer contributes to the F1 score.

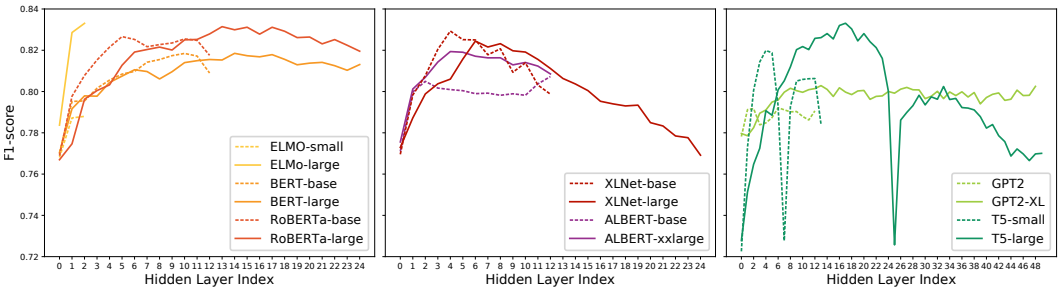


Figure 5: Hypernym predictor's F1-score across each model layer.

Figures 5-left and 5-middle show the performance across layers for the CE-based models. They reveal that while BERT and RoBERTa use their top-layers to encode semantic information, XLNet and ALBERT use the first layers. Figure 5(right) shows that GLM-based models have very distinct patterns. While GPT-2 encodes semantic information using all the layers, T5 shows an M shape because its architecture considers an encoder and decoder modules. The graph shows that T5 uses the encoder to hold most semantic information. We also note that small models show similar patterns as their larger counterparts, suggesting that these patterns depend on the architecture and not on model sizes.

7 CONCLUSIONS

In this work, we exploit the semantic conceptual taxonomy behind Wordnet to test the abilities of the current main families of pre-trained language models to learn semantic knowledge from massive sources of unlabeled data. Our main conclusion is that, indeed, to a significant extent, these models can learn relevant knowledge about the organization of concepts in Wordnet. In particular, CE models, which are based on a Transformer architecture and a contextual masking training scheme, such as BERT, appear as the more powerful family of models. We also notice that different families of models present dissimilar behavior, suggesting the encoding of different biases. Our findings also suggest that models such as XLNet and T5 encode information earlier in the pipeline. As a result, these models will greatly benefit from recent adaptive strategies to output prediction using only a limited number of blocks from the Transformer architecture. As part of future work, we hope our study helps to inspire new ideas to improve the semantic learning abilities of current pre-trained language models.

[3] http://Skylion007.github.io/OpenWebTextCorpus

REFERENCES

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? An analysis of BERT's attention. *CoRR*, abs/1906.04341, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pp. 4171–4186, 2019.

Allyson Ettinger. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the. Association of Computational Linguistics, TACL*, 8:34–48, 2020.

Yoav Goldberg. Assessing BERT's syntactic abilities. *CoRR*, abs/1901.05287, 2019.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Visualizing and understanding the effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pp. 4141–4150, 2019.

John Hewitt and Christopher D. Manning. 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, NAACL-HLT 2019*, pp. 4129–4138, 2019.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pp. 3651–3657, 2019.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of BERT. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pp. 4365–4374, 2019.

B. Lake, T. Ullman, J. Tenenbaum, and S. Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pp. 14014–14024, 2019.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26, NIPS 2013*, pp. 3111–3119, 2013.

George A. Miller. WordNet: A lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pp. 1532–1543, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018*, pp. 2227–2237, 2018.

Ofir Press, Noah A. Smith, and Omer Levy. Improving transformer models by reordering their sublayers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, pp. 2996–3005, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2018. URL `https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf`.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2019.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B. Viégas, Andy Coenen, Adam Pearce, and Been Kim. Visualizing and measuring the geometry of BERT. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pp. 8592–8600, 2019.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works. CoRR abs/2002.12327, 2020.

Chenglei Si, Shuohang Wang, Min-Yen Kan, and Jing Jiang. What does BERT learn from multiple-choice reading comprehension datasets? *CoRR*, abs/1910.12391, 2019.

Gongbo Tang, Rico Sennrich, and Joakim Nivre. An analysis of attention mechanisms: The case of word sense disambiguation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation, WMT 2018*, pp. 26–35, 2018.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pp. 4593–4601, 2019a.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019*, 2019b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems NIPS 2017*, pp. 5998–6008, 2017.

Jesse Vig. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pp. 37–42, 2019.

Elena Voita, Rico Sennrich, and Ivan Titov. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, pp. 4395–4405, 2019a.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pp. 5797–5808, 2019b.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018*, pp. 353–355, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In *Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pp. 5754–5764, 2019.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. An analysis of BERT in document ranking. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*, pp. 1941–1944, 2020.

A IMPLEMENTATION DETAILS

A.1 EDGE PROBING CLASSIFIER DETAILS

To study the extent to which these Language Models deal with semantic knowledge, we extend the methodology introduced by Tenney et al. (2019b). In that study, the authors defined a probing classifier at the sentence level, training a supervised classifier with a task-specific label. The probing classifier's motivation consists of verifying when the sentence's encoding help to solve a specific task, quantifying these results for different word embeddings models. We cast this methodology to deal with semantic knowledge extracted from WordNet. Rather than working at the sentence level, we define an edge probing classifier that learns to identify if two words are semantically related.

To create the probing classifier, we retrieve all the glosses from the Princeton WordNet Gloss Corpus[4]. The dataset provides WordNet's synsets gloss with manually matched words identifying the context-appropriate sense (see Fig 6(a)). In WordNet, each sense is coded as one of the synsets related to the word (e.g., sense *tendency.n.03* for the word tendency). Using a synet A and its specific sense provided by the tagged gloss, we retrieve from WordNet one of its direct or indirect hypernyms, denoted as B (see Fig 6(b)). If WordNet defines two or more hypernyms for A, we choose one of them at random. We sample a third synset C, at random from an unrelated section of the taxonomy, taking care that C is not related to either A or B (e.g., *animal.n.01*). Then, $\langle A, B, C\rangle$ form a triplet that allows us to create six testing edges for our classifier: $\langle A, B\rangle$, which is compounded by a pair of related words through the semantic relation *hypernym of*, and five pairs of unrelated words ($\langle A, C\rangle$, $\langle B, C\rangle$, $\langle B, A\rangle$, $\langle C, A\rangle$, $\langle C, B\rangle$). We associate a label to each of these pairs that show whether the pair is related or not (see Fig 6(c)). Note that we define directed edges, meaning that the pair $\langle A, B\rangle$ is related, but $\langle B, A\rangle$ is unrelated to the relationship *hypernym of*. Accordingly, the edge probing classifier will need to identify the pair's components and the order in which the words were declared in the pair.

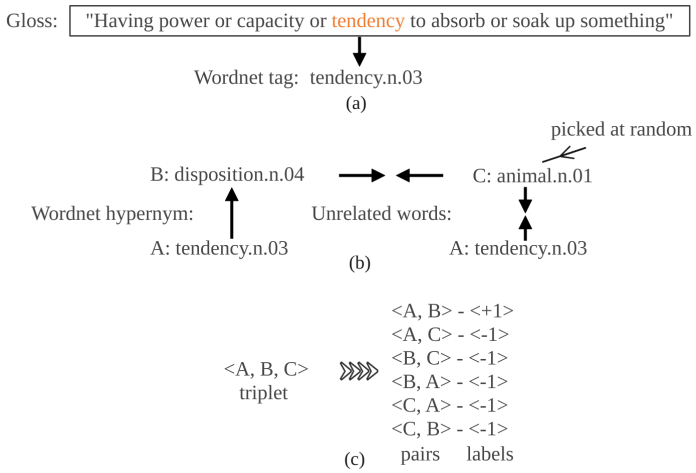


Figure 6: WordNet's synsets gloss with manually linked words to the context-appropriate sense are used to create related and unrelated pairs of words according to the relationship *hypernym of* (see figures (a) and (b)). We create six edge probing pairs, and therefore, the edge probing classifier will need to identify the pair's components and the order in which the words were declared in the pair (see figure (c)).

We create training and testing partitions ensuring that each partition has the same proportion of leaves versus internal nodes. The latter is essential to identify related pairs. During training, we guarantee that each training synset is seen at least once by the probing classifier. To guarantee the above, we sample each synset in the training set and sample some of its hypernyms at random. Then. we randomly sample some unrelated synset for each related pair that has no relation to any of the words in the related pair. We create three partitions from this data on 70/15/15 for training, development, and testing foldings, respectively.

[4] `https://wordnetcode.princeton.edu/glosstag.shtml`

We train the MLP classifier using a weighted binary cross-entropy loss function. Since we have one positive and five negative examples per triplet, we use a weighted loss function with weights 5 and 1 for the positive and negative class, respectively. Accordingly, positive and negative examples have the same relevance during training. We implemented the linear layer and the MLP classifier using FFNN with 384 hidden units. The MLP was trained using dropout at 0.425 and a L_2 regularizer to avoid overfitting.

To create the vector representations for each of the word embeddings models considered in this study, we concatenate the hidden state vectors of all the layers for each tagged synset. For both CE and GLM-based models, each gloss was used as a context to build specific contextual word embeddings. If the gloss has more than one tagged token, we take only the first of them for the analysis.

A.2 WORDNET METRICS: DISTANCE

Formally, let $d_W(x, y)$ be the Wordnet distance between two synsets x, y, defined by:

$$d_W(x,y) = \begin{cases} d_{\text{path}}(x,y) & \text{if } y \text{ is ancestor of } x, \\ d_{\text{path}}(x,z) + d_{\text{path}}(y,z) & \text{otherwise,} \end{cases}$$

where $d_{\text{path}}(x, y)$ is the length of the shortest path between x and y in WordNet, measured in number of hops, and z is the closest common ancestor of x and y in the case that y is not an ancestor of x.

B RECONSTRUCTED GRAPHS

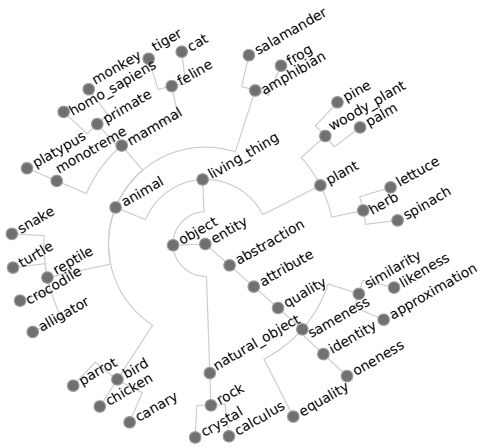
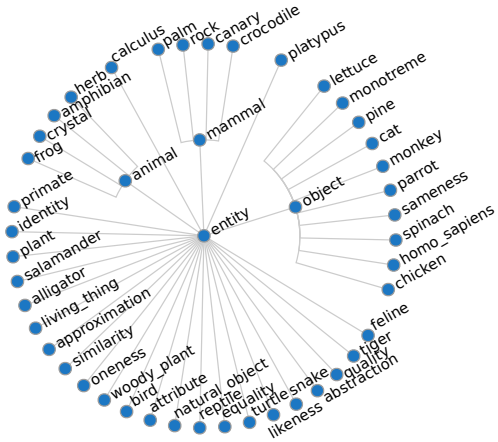
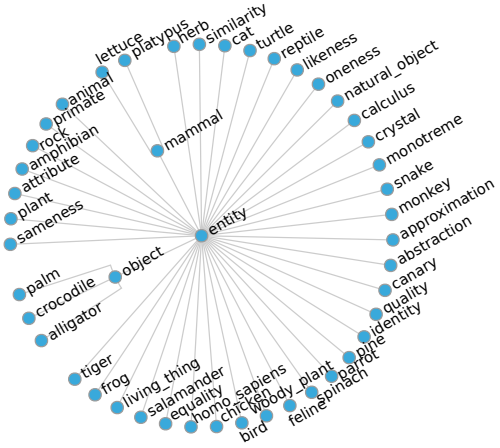


Figure 7: Ground Truth Knowledge Graph

(a) Word2Vec reconstruction using MCM



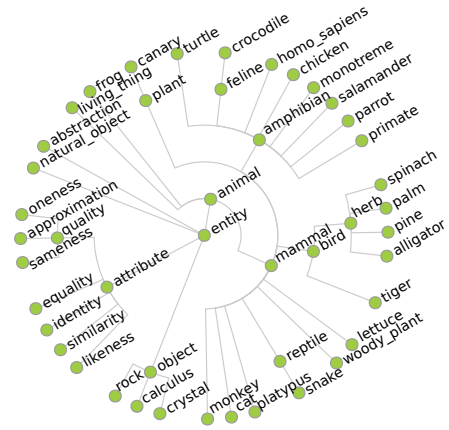
(b) Word2Vec reconstruction using TIM



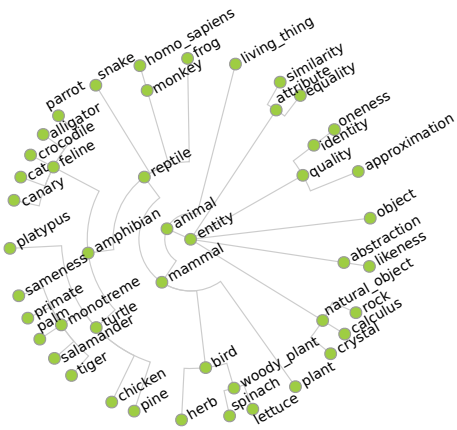
(c) GloVe-42B reconstruction using MCM

(d) GloVe-42B reconstruction using TIM

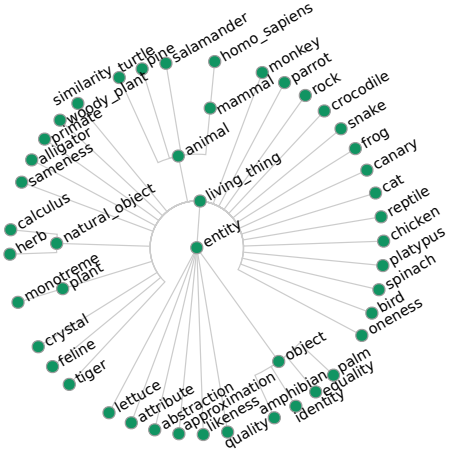
Figure 8: Knowledge graph reconstruction using Word2Vec and GloVe.



(a) GPT-2-XL reconstruction using MCM



(b) GPT-2-XL reconstruction using TIM



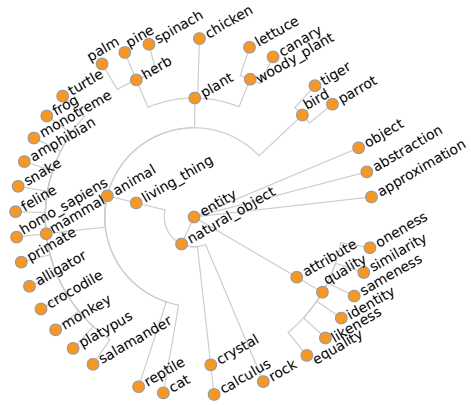
(c) T5-large reconstruction using MCM

(d) T5-large reconstruction using TIM

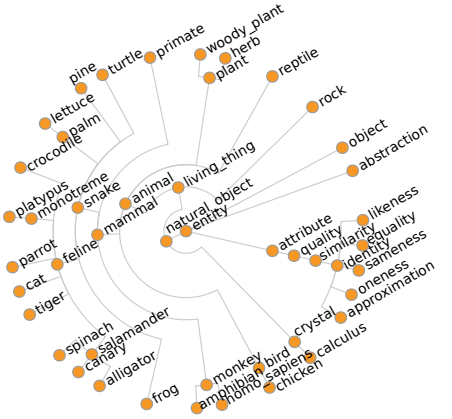
Figure 9: Knowledge graph reconstruction using GPT-2 and T5.

(a) ELMo-large reconstruction using MCM

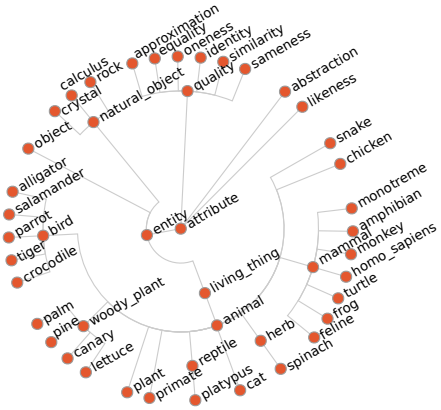
(b) ELMo-large reconstruction using TIM



(c) BERT-large reconstruction using MCM



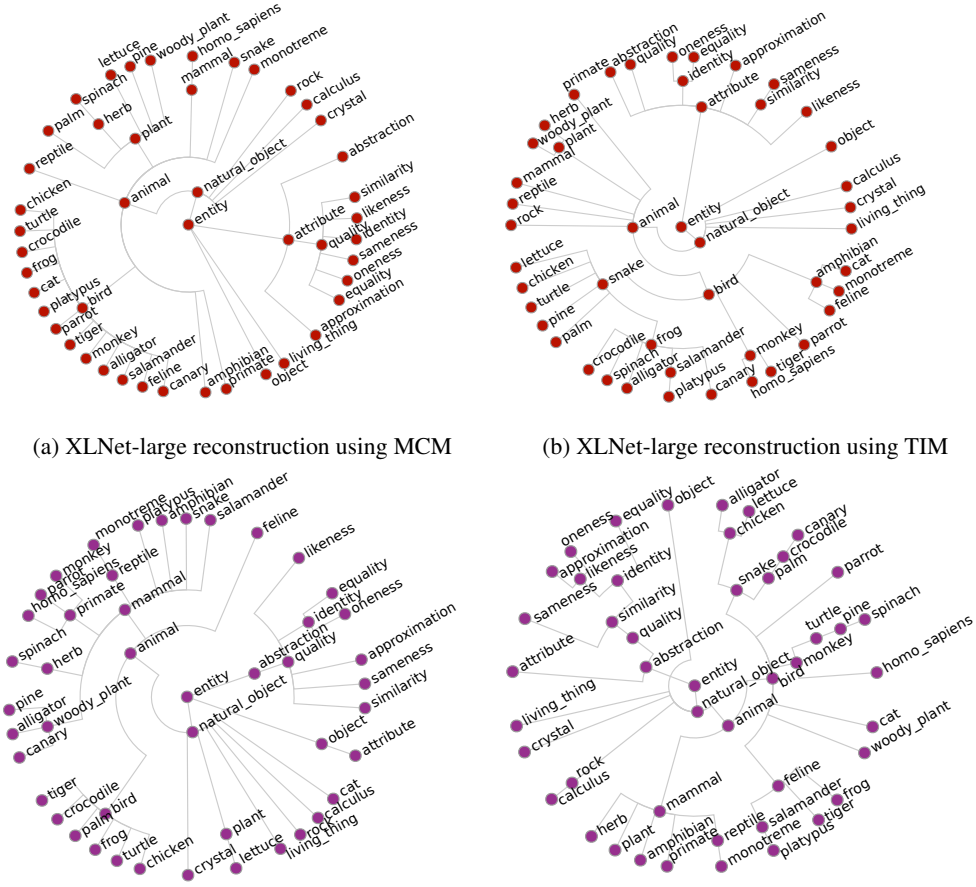
(d) BERT-large reconstruction using TIM



(e) RoBERTa-large reconstruction using MCM

(f) RoBERTa-large reconstruction using TIM

Figure 10: Knowledge graph reconstruction using ELMo, BERT and RoBERTa.



(a) XLNet-large reconstruction using MCM

(b) XLNet-large reconstruction using TIM

(c) ALBERT-large reconstruction using MCM

(d) ALBERT-large reconstruction using TIM

Figure 11: Knowledge graph reconstruction using XLNet and ALBERT.

C FURTHER ANALYSIS OF FACTORS' IMPACT IN MODELS PERFORMANCE

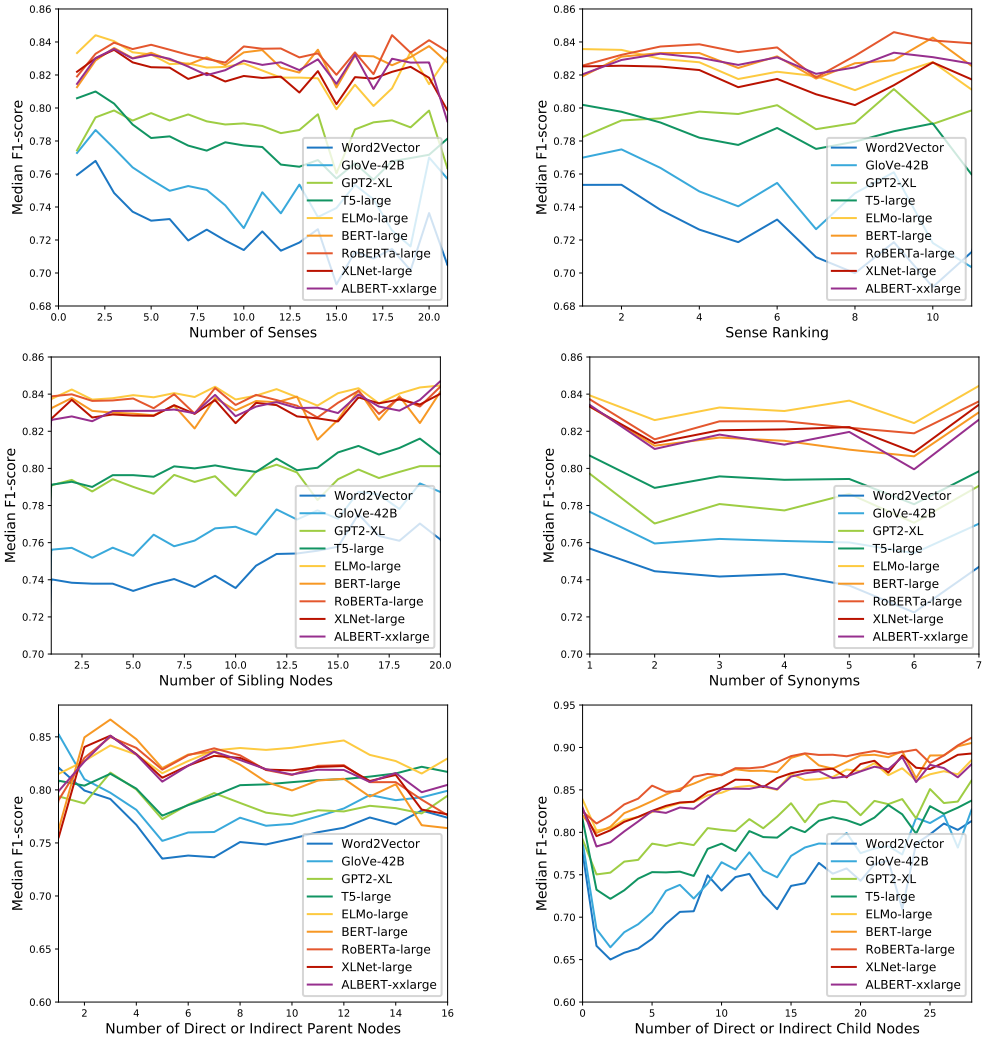


Figure 12: Many WordNet factors were analyzed to elucidate how they affect these models' performance in the semantic probing task. We considered among these factors the number of senses, the sense ranking, the number of sibling nodes, the number of synonyms, the number of direct or indirect parent nodes, and the number of direct and indirect child nodes.