Contextual Embeddings: When Are They Worth It?

Simran Arora[*] **Avner May**[*] **Jian Zhang, Christopher Ré**
Stanford Univeristy
{simarora, avnermay, zjian, chrismre}@cs.stanford.edu

Abstract

We study the settings for which deep contextual embeddings (e.g., BERT) give large improvements in performance relative to classic pretrained embeddings (e.g., GloVe), and an even simpler baseline—*random* word embeddings—focusing on the impact of the training set size and the linguistic properties of the task. Surprisingly, we find that *both* of these simpler baselines can *match* contextual embeddings on industry-scale data, and often perform within 5 to 10% accuracy (absolute) on benchmark tasks. Furthermore, we identify properties of data for which contextual embeddings give particularly large gains: language containing complex structure, ambiguous word usage, and words unseen in training.

1 Introduction

In recent years, rich contextual embeddings such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) have enabled rapid progress on benchmarks like GLUE (Wang et al., 2019a) and have seen widespread industrial use (Pandu Nayak, 2019). However, these methods require significant computational resources (memory, time) during pretraining, and during downstream task training and inference. Thus, an important research problem is to understand when these contextual embeddings add significant value vs. when it is possible to use more efficient representations without significant degradation in performance.

As a first step, we empirically compare the performance of contextual embeddings with classic embeddings like word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). To further understand what performance gains are attributable to improved embeddings vs. the powerful downstream models that leverage them, we also compare with a simple baseline—*fully random embeddings*—which encode no semantic or contextual information whatsoever. Surprisingly, we find that in highly optimized production tasks at a major technology company, *both* classic and random embeddings have competitive (or even slightly better!) performance than the contextual embeddings.[1,2]

To better understand these results, we study the properties of NLP tasks for which contextual embeddings give large gains relative to non-contextual embeddings. In particular, we study how the amount of training data, and the linguistic properties of the data, impact the relative performance of the embedding methods, with the intuition that contextual embeddings should give limited gains on data-rich, linguistically simple tasks.

In our study on the impact of training set size, we find in experiments across a range of tasks that the performance of the non-contextual embeddings (GloVe, random) improves rapidly as we increase the amount of training data, often attaining within 5 to 10% accuracy of BERT embeddings when the full training set is used. This suggests that for many tasks these embeddings could likely match BERT given sufficient data, which is precisely what we observe in our experiments with industry-scale data. Given the computational overhead of contextual embeddings, this exposes important trade-offs between the computational resources required by the embeddings, the expense of labeling training data, and the accuracy of the downstream model.

To better understand when contextual embeddings give large boosts in performance, we identify three linguistic properties of NLP tasks which help explain when these embeddings will provide gains:

- **Complexity of sentence structure**: How interdependent are different words in a sentence?

[*] Equal contribution.

[1] This aligns with recent observations from experiments with classic word embeddings at Apple (Ré et al., 2020).

[2] These tasks are proprietary, so we share these results anecdotally as motivation for our study.

- **Ambiguity in word usage**: Are words likely to appear with multiple labels during training?
- **Prevalence of unseen words**: How likely is encountering a word never seen during training?

Intuitively, these properties distinguish between NLP tasks involving simple and formulaic text (e.g., assistant commands) vs. more unstructured and lexically diverse text (e.g., literary novels). We show on both sentiment analysis and NER tasks that contextual embeddings perform significantly better on more complex, ambiguous, and unseen language, according to proxies for these properties. Thus, contextual embeddings are likely to give large gains in performance on tasks with a high prevalence of this type of language.

2 Background

We discuss the different types of word embeddings we compare in our study: contextual pretrained embeddings, non-contextual pretrained embeddings, and random embeddings; we also discuss the relative efficiency of these embedding methods, both in terms of computation time and memory (Sec. 2.1).

Pretrained contextual embeddings Recent contextual word embeddings, such as BERT (Devlin et al., 2018) and XLNet (Yang et al., 2019), consist of multiple layers of transformers which use self-attention (Vaswani et al., 2017). Given a sentence, these models encode each token into a feature vector which incorporates information from the token's context in the sentence.

Pretrained non-contextual embeddings Non-contextual word embeddings such as GloVe (Pennington et al., 2014), word2vec (Mikolov et al., 2013), and fastText (Mikolov et al., 2018) encode each word in a vocabulary as a vector; intuitively, this vector is meant to encode semantic information about a word, such that similar words (e.g., synonyms) have similar embedding vectors. These embeddings are pretrained from large language corpora, typically using word co-occurrence statistics.

Random embeddings In our study, we consider random embeddings (e.g., as in Limsopatham and Collier (2016)) as a simple and efficient baseline that requires no pretraining. Viewing word embeddings as n-by-d matrices (n: vocabulary size, d: embedding dimension), we consider embedding matrices composed entirely of random values. To reduce the memory overhead of storing these $n \cdot d$ random values to $O(n)$, we use circulant random matrices (Yu et al., 2017) as a simple and efficient approach (for more details, see Appendix A.1).[3,4]

2.1 System Efficiency of Embeddings

We discuss the computational and memory requirements of the different embedding methods, focusing on downstream task training and inference.[5]

Computation time For deep contextual embeddings, extracting the word embeddings for tokens in a sentence requires running inference through the full network, which takes on the order of 10 ms on a GPU. Non-contextual embeddings (e.g., GloVe, random) require negligible time ($O(d)$) to extract an embedding vector.

Memory Using contextual embeddings for downstream training and inference requires storing all the model parameters, as well as the model activations during training if the embeddings are being fine-tuned (e.g., 440 MB to store $\text{BERT}_{\text{BASE}}$ parameters, and on the order of 5-10 GB to store activations). Pretrained non-contextual embeddings (e.g., GloVe) require $O(nd)$ to store a n-by-d embedding matrix (e.g., 480 MB to store a 400k by 300 GloVe embedding matrix). Random embeddings take $O(1)$ memory if only the random seed is stored, or $O(n)$ if circulant random matrices are used (e.g., 1.6 MB if $n = 400\text{k}$).

3 Experiments

We provide an overview of our experimental protocols (Section 3.1), the results from our study on the impact of training set size (Section 3.2), and the results from our linguistic study (Section 3.3). We show that the gap between contextual and non-contextual embeddings often shrinks as the amount of data increases, and is smaller on language that is *simpler* based on linguistic criteria we identify.

3.1 Experimental Details

To study the settings in which contextual embeddings give large improvements, we compare

[3]Note that one could also simply store the random seed, though this requires regenerating the embedding matrix every time it is accessed.

[4]We provide an efficient implementation of circulant random embedding matrices here: https://github.com/HazyResearch/random_embedding.

[5]Pretrained contextual and non-contextual embeddings also require significant computational resources during pretraining. For example training $\text{BERT}_{\text{BASE}}$ takes 4 days on 16 TPU chips.

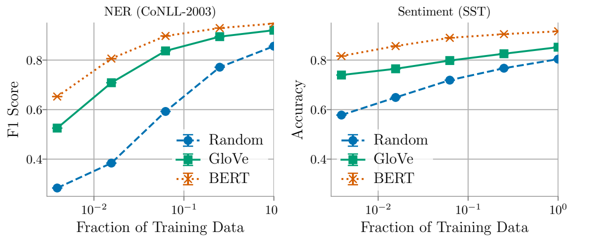

Figure 1: NER (CoNLL-2003; left), and sentiment analysis (SST; right) performance, as a function of the fraction of the training set used. As the amount of training data increases, the non-contextual embedding performance improves quickly, generally narrowing the gap with the contextual embeddings.

them to GloVe and random embeddings across a range of named entity recognition (NER) (Tjong Kim Sang and De Meulder, 2003), sentiment analysis (Kim, 2014), and natural language understanding (Wang et al., 2019a) tasks. We choose these lexically diverse tasks as examples of word, sentence, and sentence-pair classification tasks, respectively. For our embeddings, we consider 768-dimensional pretrained $\text{BERT}_{\text{BASE}}$ word embeddings, 300-dimensional publicly available GloVe embeddings, and 800-dimensional random circulant embeddings. We keep the embedding parameters fixed during training for all embedding types (no fine-tuning), to isolate the benefits of pretraining from the benefits of task training. We use a CNN model (Kim, 2014) for sentiment analysis and a BiLSTM (Akbik et al., 2018; Wang et al., 2019a) for the NER and General Language Understanding Evaluation (GLUE) tasks. For more details on the tasks, models, and training protocols, please see Appendix A.

3.2 Impact of Training Data Volume

We show that the amount of downstream training data is a critical factor in determining the relative performance of contextual vs. non-contextual embeddings. In particular, we show in representative tasks in Figure 1 that the performance of the non-contextual embedding models improves quickly as the amount of training data is increased (plots for all tasks in Appendix B).[6] As a result of this improvement, we show in Table 1 that across tasks when the full training set is used, the non-contextual embeddings can often (1) perform within 10% absolute accuracy of the contextual embeddings, and (2) match the performance of the contextual embeddings trained on 1x-16x less data, while also being orders of magnitude more computationally efficient. In light of this, ML practitioners may find that for certain real-world tasks the large gains in efficiency are well worth the cost of labeling more data.

Specifically, in this table we show for each task the difference between the accuracies attained by BERT vs. GloVe and random (note that random sometimes beats GloVe!), as well as the largest integer $n \in \{1, 4, 16, 64, 256\}$ such that BERT trained on $\frac{1}{n}$ of the training set still outperforms non-contextual embeddings trained on the full set.

[6]We provide theoretical support for why random embeddings perform strongly given sufficient data in Appendix B.3.

Task		Performance gap			Sample complexity ratio	
		B	R-B	G-B	R/B	G/B
NER	CoNLL	94.8	-9.1	-2.7	16	4
Sent.	TREC	95.8	-10.3	-6.0	4	4
	MPQA	89.6	-4.7	-0.9	16	1
	CR	88.5	-5.0	-3.5	16	4
	SUBJ	97.7	-8.7	-3.3	256	16
	SST	91.6	-11.2	-6.4	256	64
	MR	85.9	-13.4	-6.9	256	16
GLUE	RTE	61.0	-1.8	-2.5	1	64
	MRPC	84.8	-8.8	-6.9	16	4
	QQP	86.5	-9.2	-6.8	16	16
	CoLA	51.7	-34.6	-40.1	64	256
	STS-B	85.6	-29.1	-19.8	64	16
	QNLI	84.6	-15.8	-9.5	64	16
	MNLI	78.8	-17.1	-12.0	64	16
	SST	91.3	-15.9	-12.8	256	64

Table 1: Performance and sample complexity of random (R) and GloVe (G) relative to BERT (B) for NER, sentiment analysis (Sent.), and language understanding (GLUE) tasks. Second column shows BERT accuracy; third/fourth columns show the accuracy gap between BERT and random/GloVe; fifth/sixth columns show *sample complexity ratios*, the largest $n \in \{1, 4, 16, 64, 256\}$ for which BERT outperforms random/GloVe when trained on n-times less data. We observe that non-contextual embeddings can often (1) perform within 10% absolute accuracy of the contextual embeddings, and (2) match the performance of contextual embeddings which are trained on 1x-16x less data. This sheds light on a tradeoff between the upfront cost of labeling training data and the inference-time computational cost of the embeddings.

3.3 Study of Linguistic Properties

In this section, we aim to identify properties of the language in a dataset for which contextual embeddings perform particularly well relative to non-contextual approaches. Identifying such properties would allow us to determine whether a new task is likely to benefit from contextual embeddings.

As a first step in our analysis, we evaluate the different embedding types on the GLUE Diagnostic Dataset (Wang et al., 2019a). This task defines four categories of linguistic properties; we observe that the contextual embeddings performed similarly to the non-contextual embeddings for three categories, and significantly better for the *predicate-argument structure* category (Matthews correlation coefficients of .33, .20, and .20 for BERT, GloVe, and random, respectively. See Appendix C.2.1 for more detailed results). This category requires understanding how sentence subphrases are composed together (e.g., prepositional phrase attachment, and identifying a verb's subject and object). Motivated by the observation that contextual embeddings are systematically better on specific types of linguistic phenomena, we work to identify simple and quantifiable properties of a downstream task's language which correlate with large boosts in performance from contextual embeddings.

In the context of both word-level (NER) and sentence-level (sentiment analysis) classification tasks, we define metrics that measure (1) the complexity of text structure, (2) the ambiguity in word usage, and (3) the prevalence of unseen words (Section 3.3.1), and then show that contextual embeddings attain significantly higher accuracy than non-contextual embeddings on inputs with high metric values (Section 3.3.2, Table 2).

3.3.1 Metric Definitions

We now present our metric definitions for NER and sentiment analysis, organized by the above three properties (detailed definitions in Appendix C).

Complexity of text structure We hypothesize that language with more complex internal structure will be harder for non-contextual embeddings. We define the metrics as follows:

- **NER**: We consider the number of tokens spanned by an entity as its complexity metric (e.g., "George Washington" spans 2 tokens), as correctly labeling a longer entity requires understanding the *relationships* between the different tokens in the entity name.
- **Sentiment analysis**: We consider the average distance between pairs of dependent tokens in a sentence's dependency parse as a measure of the sentence's complexity, as long-range dependencies are typically a challenge for NLP systems.

Ambiguity in word usage We hypothesize that non-contextual embeddings will perform poorly in disambiguating words that are used in multiple different ways in the training set. We define the metrics as follows:

- **NER**: We consider the number of labels (person, location, organization, miscellaneous, other) a token appears with in the training set as a measure of its ambiguity (e.g., "Washington" appears as a person, location, and organization in CoNLL-2003).
- **Sentiment analysis**: As a measure of a sentence's ambiguity, we take the average over the words in the sentence of the probability that the word is positive in the training set, and compute the entropy of a coin flip with this probability.[7]

Prevalence of unseen words We hypothesize that contextual embeddings will perform significantly better than non-contextual embeddings on words which do not appear at all in the training set for the task. We define the following metrics:

- **NER**: For a token in the NER input, we consider the inverse of the number of times it was seen in the training set (letting $1/0 := \infty$).
- **Sentiment analysis**: Given a sentence, we consider as our metric the fraction of words in the sentence that were never seen during training.

3.3.2 Empirical validation of metrics

In Table 2 we show that for each of the metrics defined above, the accuracy gap between BERT and random embeddings is larger on inputs for which the metrics are large. In particular, we split each of the task validation sets into two halves, with points with metric values below the median in one half, and above the median in the other. We see that in 19 out of 21 cases, the accuracy gap between BERT and random embeddings is larger on the slice of the validation set corresponding to large metric values, validating our hypothesis that contextual embeddings provide important boosts in accuracy on these points.

In Appendix C.2.2, we present a similar table comparing the performance of BERT and GloVe embeddings. We see that the gap between GloVe and BERT errors is larger above the median than below it in 11 out of 14 of the complexity and am-

[7] For sentiment tasks with C-labels ($C = 6$ for the TREC dataset), we consider the entropy of the average label distribution $\frac{1}{n}\sum_{i=1}^{n} p(y|w_i) \in \mathbb{R}^C$ over the sentence words w_i.

	Complexity		Ambiguity		Unseen	
Task	Abs.	Rel.	Abs.	Rel.	Abs.	Rel.
NER (CoNLL)	+4.6	1.4	+7.7	2.0	+5.0	1.4
Sent. (MR)	-5.4	0.7	+3.3	1.3	+1.2	1.1
Sent. (SUBJ)	-1.8	0.8	+6.7	2.3	+0.9	1.1
Sent. (CR)	+0.6	1.1	+3.0	1.8	+4.1	2.4
Sent. (SST)	+7.4	2.1	+8.7	2.4	+2.3	1.2
Sent. (TREC)	+5.1	1.7	+5.9	1.8	+4.4	1.5
Sent. (MPQA)	+7.9	13.5	+7.1	12.4	+1.3	1.4

Table 2: For our complexity, ambiguity, and unseen prevalence metrics, we slice the validation set using the median metric value, and compute the average error rates for BERT and random on each slice. We show that the gap between BERT and random errors is larger on the slice above the median than below it in 19 out of 21 cases, in absolute (Abs.) and relative (Rel.) terms.

biguity results, which is consistent with our hypothesis that context is helpful for structurally complex and ambiguous language. However, we observe that GloVe and BERT embeddings—which can both leverage pretrained knowledge about unseen words—perform relatively similarly to one another above and below the median for the unseen metrics.

4 Related Work

The original work on ELMo embeddings (Peters et al., 2018) showed that the gap between contextual and non-contextual embeddings narrowed as the amount of training data increased. Our work builds on these results by additionally comparing with random embeddings, and by studying the linguistic properties of tasks for which the contextual embeddings give large gains.

Our work is not the first to study the downstream performance of embeddings which do not require any pretraining. For example, in the context of neural machine translation (NMT) it is well-known that randomly-initialized embeddings can attain strong performance (Wu et al., 2016; Vaswani et al., 2017); the work of Qi et al. (2018) empirically compares the performance of pretrained and randomly-initialized embeddings across numerous languages and dataset sizes on NMT tasks, showing for example that the pretrained embeddings typically perform better on similar language pairs, and when the amount of training data is small (but not too small). Furthermore, as mentioned in Section 2, random embeddings were considered as a baseline by Limsopatham and Collier (2016), to better understand the gains from using generic vs. domain-specific word embeddings for text classification tasks. In contrast, our goal for using random embeddings in our study was to help clarify when and why pretraining gives gains, and to expose an additional operating point in the trade-off space between computational cost, data-labeling cost, and downstream model accuracy.

5 Conclusion

We compared the performance of contextual embeddings with non-contextual pretrained embeddings and with an even simpler baseline—random embeddings. We showed that these non-contextual embeddings perform surprisingly well relative to the contextual embeddings on tasks with plentiful labeled data and simple language. While much recent and impressive effort in academia and industry has focused on improving state-of-the-art performance through more sophisticated, and thus increasingly expensive, embedding methods, this work offers an alternative perspective focused on realizing the trade-offs involved when choosing or designing embedding methods. We hope this work inspires future research on better understanding the differences between embedding methods, and on designing simpler and more efficient models.

Acknowledgments

We gratefully acknowledge the support of DARPA under Nos. FA87501720095 (D3M), FA86501827865 (SDH), and FA86501827882 (ASED); NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); ONR under No. N000141712266 (Unifying Weak Supervision); the Moore Foundation, NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, the Okawa Foundation, American Family Insurance, Google Cloud, Swiss Re, the Stanford Graduate Fellowship in Science and Engineering, and members of the Stanford DAWN project: Teradata, Facebook, Google, Ant Financial, NEC, VMWare, and Infosys. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, NIH, ONR, or the U.S. Government.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: an easy-to-use framework for state-of-the-art NLP. In *NAACL-HLT (Demonstrations)*, pages 54–59. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING*.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Nut Limsopatham and Nigel Collier. 2016. Modelling the combination of generic and target domain embeddings in a convolutional neural network for sentence classification. In *BioNLP@ACL*, pages 136–140. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *LREC*.

Pandu Nayak. 2019. Understanding searches better than ever before. [Online; published 25-Oct-2019; accessed 6-Dec-2019].

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *NAACL-HLT (2)*, pages 529–535. Association for Computational Linguistics.

Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press.

Christopher Ré, Feng Niu, Pallavi Gudipati, and Charles Srisuwananukorn. 2020. Overton: A data system for monitoring and improving machine-learned products. In *CIDR*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *NAACL-HLT*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

Alex Wang, Ian F. Tenney, Yada Pruksachatkun, Katherin Yu, Jan Hula, Patrick Xia, Raghu Pappagari, Shuning Jin, R. Thomas McCoy, Roma Patel, Yinghui Huang, Jason Phang, Edouard Grave, Haokun Liu, Najoung Kim, Phu Mon Htut, Thibault Févry, Berlin Chen, Nikita Nangia, Anhad Mohananey, Katharina Kann, Shikha Bordia, Nicolas Patry, David Benton, Ellie Pavlick, and Samuel R. Bowman. 2019b. `jiant` 1.2: A software toolkit for research on general-purpose text understanding models. http://jiant.info/.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Felix X. Yu, Aditya Bhaskara, Sanjiv Kumar, Yunchao Gong, and Shih-Fu Chang. 2017. On binary embedding using circulant matrices. *JMLR*, 18(1):5507–5536.

Felix X Yu, Sanjiv Kumar, Henry Rowley, and Shih-Fu Chang. 2015. Compact nonlinear maps and circulant extensions. *arXiv preprint arXiv:1503.03893*.

A Experimental Details

We now describe the embeddings (Appendix A.1), tasks (Appendix A.2), and models (Appendix A.3) we use in our experiments in more detail.

A.1 Embeddings

We compare the performance of BERT contextual embeddings with GloVe embeddings and random embeddings. We specifically use 768-dimensional $\text{BERT}_{\text{BASE}}$ WordPiece embeddings, 300-dimensional GloVe embeddings, and 800-dimensional random embeddings. We freeze each set of embeddings prior to training, and do not fine-tune the embeddings during training. The random embeddings are normalized to have the same Frobenius norm as the GloVe embeddings. We now describe how we use circulant matrices to reduce the memory requirement for the random embeddings.

Circulant Random Embeddings To store a random n-by-d matrix in $O(n)$ memory instead of $O(nd)$, we use random circulant matrices (Yu et al., 2017). Specifically, we split the n-by-d matrix into $\frac{n}{d}$ disjoint d-by-d sub-matrices (assuming for simplicity that d divides n evenly), where each sub-matrix is equal to CD, where $C = \text{circ}(c) \in \mathbb{R}^{d\times d}$ is a circulant matrix based on a random Gaussian vector $c \in \mathbb{R}^d$, and $D = \text{diag}(r) \in \mathbb{R}^{d\times d}$ is a diagonal matrix based on a random Radamacher vector $r \in \{-1, +1\}^d$. Note that a circulant matrix $\text{circ}(c)$ is defined as follows:

$$\text{circ}(c) := \begin{pmatrix} c_0 & c_d & \dots & c_2 & c_1 \\ c_1 & c_0 & \dots & c_3 & c_2 \\ \dots & \dots & \ddots & \dots & \dots \\ c_{d-1} & c_{d-2} & \dots & c_0 & c_d \\ c_d & c_{d-1} & \dots & c_1 & c_0 \end{pmatrix}.$$

Random circulant embeddings have been used in the kernel literature to make kernel approximation methods more efficient (Yu et al., 2015). For downstream training and inference, one can simply store the d-dimensional c and r vectors for each of the $\frac{n}{d}$ disjoint d-by-d sub-matrices, taking a total of $O(n)$ memory. Alternatively, one can simply store a single random seed ($O(1)$ memory), and these c, r vectors can be regenerated on the fly each time a row of the embedding matrix is accessed. Note that in addition to being very memory efficient, random embeddings avoid the expensive pretraining process over a large language corpus.

A.2 Tasks

We perform evaluations on three types of standard downstream NLP tasks: named entity recognition (NER), sentiment analysis, and natural language understanding. NER involves classifying each token in the input text as an entity or a non-entity, and further classifying the entity type for identified entities. We evaluate on the CoNLL-2003 benchmark dataset, in which each token is assigned a label of “O” (non-entity), “PER” (person), “ORG” (organization), “LOC” (location), or “MISC” (miscellaneous). Sentiment analysis involves assigning a classification label at the sentence level corresponding to the sentiment of the sentence. We evaluate on five binary sentiment analysis benchmark datasets including MR, MPQA, CR, SST, and SUBJ. We also evaluate on the benchmark TREC dataset, which assigns one of six labels to each input example. For natural language understanding, we use the standard GLUE benchmark tasks, and the GLUE diagnostic task.

A.3 Downstream Task Models

We use the following models and training protocols for the NER, sentiment analysis, and GLUE tasks:

NER: We use a BiLSTM task model with a CRF decoding layer, and we use the default hyperparameters from the flair (Akbik et al., 2019) repository:[8] 256 hidden units, 32 batch size, 150 max epochs, and a stop-condition when the learning rate decreases below 0.0001 with a decay constant of 0.5 and patience of 4. In our evaluation, we report micro-average F1-scores for this task.

Sentiment analysis: We use the architecture and training protocol from Kim (2014), using a CNN with 1 convolutional layer, 3 kernel sizes in $\{3, 4, 5\}$, 100 kernels, 32 batch size, 100 max epochs, and a constant learning rate. We report the validation error rates in evaluations of each task.

GLUE: We use the Jiant (Wang et al., 2019b) implementation of a BiLSTM with 1024 hidden dimensions, 2 layers, 32 batch size, and a stop-condition when the learning rate decreases below 0.000001 with a decay constant of 0.5 and patience of 5. We consider the following task-specific performance metrics: Matthews correlation for CoLA, MNLI, and the diagnostic task, validation F1-score for MRPC and QQP, and validation accuracy for QNLI and RTE.

[8] https://github.com/zalandoresearch/flair.

B Impact of Training Data Volume

We now provide additional details regarding our experiments on the impact of training set size on performance (Appendix B.1), our complete set of empirical results from these experiments (Appendix B.2), as well as theoretical support for the strong performance of random embedding models in these experiments, when trained with sufficient downstream data (Appendix B.3).

B.1 Additional Experiment Details

For each task, we evaluate performance using five fractions of the full training dataset, to understand how the amount of training data affects performance: $\{\frac{1}{4^4}, \frac{1}{4^3}, \frac{1}{4^2}, \frac{1}{4^1}, 1\}$. For each fraction c, we randomly select a subset of the training set of the corresponding size, and replicate this data $1/c$ times; we then train models using this redundant dataset, using the model architectures and training protocols described in Appendix A.3. In downstream training we perform a seperate hyperparameter sweep of the learning rate at each fraction of the training data, and select the best learning rate for each embedding type. We use the following lists of learning rates for the different tasks:

- **NER**: $\{.003, .01, .03, .1, .3, 1, 3\}$.
- **Sentiment analysis**: {1e-5, 3e-5, 1e-4, 3e-4, 1e-3, 3e-2, 1e-2}.
- **GLUE**: {1e-6, 3e-6, 1e-5, 3e-5, 1e-4, 3e-4, 1e-3}.

B.2 Extended Results

In Figures 2 and 3, we show the performance of random, GloVe, and BERT embeddings on all the NER, sentiment analysis, and GLUE tasks, as we vary the amount of training data. We can see that across most of these results:

- Non-contextual embedding performance improves quickly as the amount of training data is increased.
- The gap between contextual and non-contextual embeddings often shrinks as the amount of training data is increased.
- There are many tasks for which random and GloVe embeddings perform relatively similarly to one another.

B.3 Theoretical Support for Random Embedding Performance

To provide theoretical support for why, given sufficient training data, a model trained with random embeddings might match the performance of one trained with pretrained embeddings, we consider the simple setting of Gaussian process (GP) regression (Rasmussen and Williams, 2006). In particular, we assume that the prior covariance function for the GP is determined by the pretrained embeddings, and show that as the number of observed samples from this GP grows, the posterior distribution gives diminishing weight to the prior covariance function, and eventually depends solely on the observed samples. Thus, if we were to calculate the posterior distribution using an *inaccurate* prior covariance function determined by random embeddings, this posterior would approach the true posterior as the number of observed samples grew.

More formally, for a fixed set of words $\{w_1, \ldots, w_n\}$ with pretrained embeddings $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, we assume that the "true" regression label vector $y^* \in \mathbb{R}^n$ for these words is sampled from a zero-mean multivariate Gaussian distribution $y^* \sim \mathcal{N}(0, K)$, where the entries $K_{ij} := k(x_i, x_j)$ of the covariance matrix K are determined based on the similarity $k(x_i, x_j)$ between the pretrained embeddings $x_i, x_j \in \mathbb{R}^d$ for words i and j.[9] We then assume that we observe m noisy samples $(y_1, \ldots, y_m)$ of the "true" label vector y^*, where each $y_i \in \mathbb{R}^n$ is an independent sample from $\mathcal{N}(y^*, \sigma^2 I)$. To summarize:

$$
\begin{aligned}
y^* &\sim \mathcal{N}(0, K), \\
y_1, \ldots, y_m &\sim \mathcal{N}(y^*, \sigma^2 I).
\end{aligned}
$$

The question then becomes, what is the posterior distribution for y^* after observing $(y_1, \ldots, y_m)$? The closed form solution for this posterior is as follows:

$$
\begin{aligned}
p(y^* \mid y_1, \ldots, y_m) &= \mathcal{N}(\bar{y}_m, \bar{K}_m), \text{ where} \\
\bar{y}_m &= K\left(K + \frac{\sigma^2}{m} I\right)^{-1} \left(\frac{1}{m}\sum_{i=1}^m y_i\right), \\
\bar{K}_m &= K\left(K + \frac{\sigma^2}{m} I\right)^{-1} \frac{\sigma^2}{m} I.
\end{aligned}
$$

Importantly, we observe that as $m \to \infty$, that $\bar{y}_m \to y^*$ (because $K(K + \frac{\sigma^2}{m} I)^{-1} \to I$ and

[9] As an example, we could have $k(x_i, x_j) := \exp\left(-\|x_i - x_j\|^2 / (2\sigma^2)\right)$ be the Gaussian kernel.

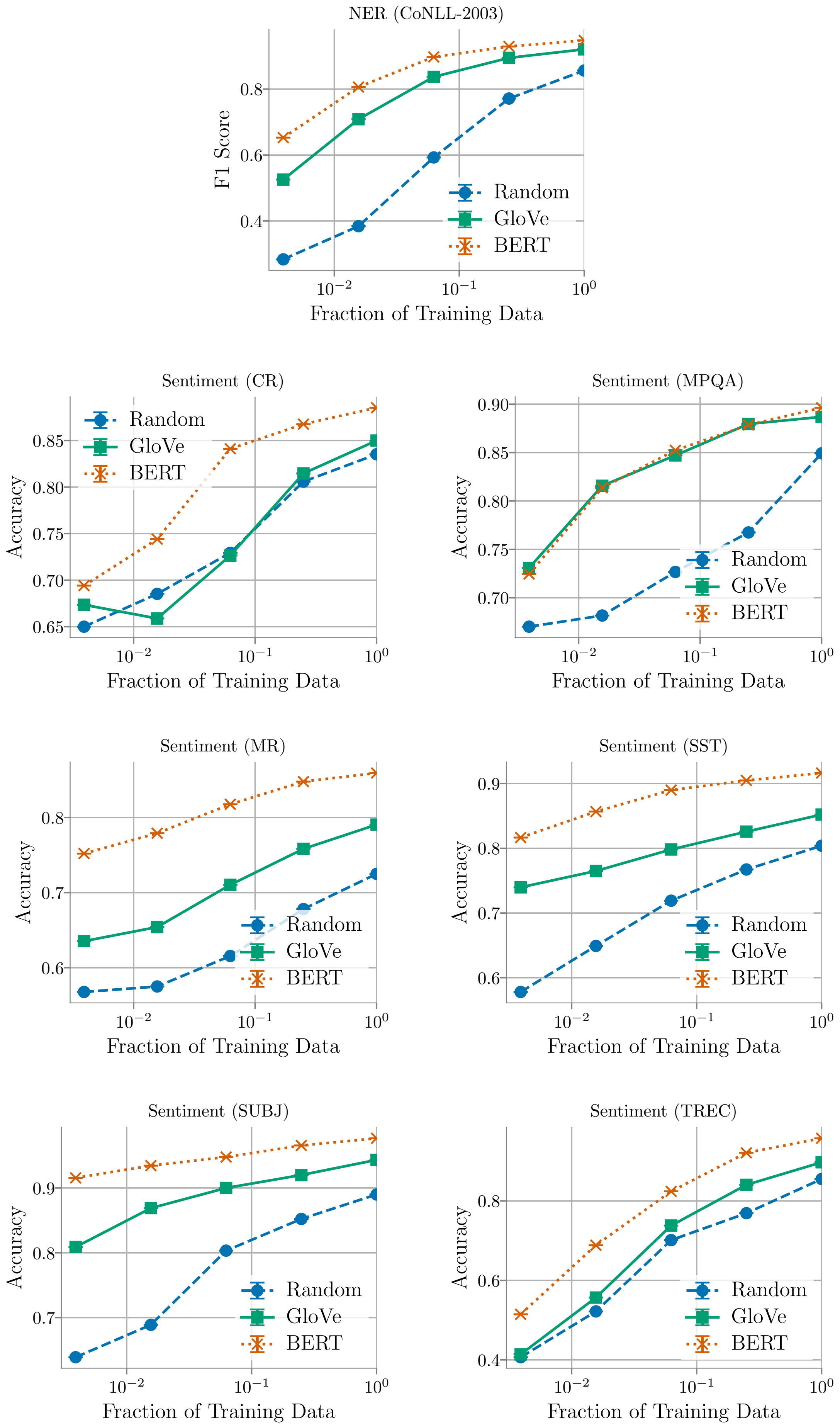

Figure 2: Performance of random, GloVe, and BERT embeddings on the NER (top row) and sentiment analysis (bottom three rows) tasks as we vary the amount of training data

Figure 3: Performance of random, GloVe, and BERT embeddings on GLUE tasks as we vary the amount of training data

$\frac{1}{m}\sum_{i=1}^{m} y_i \to y^*$), and $\bar{K}_m \to 0$. Thus, if we were to compute the posterior distribution for this GP using an uninformative prior covariance function K' determined by random embeddings $\{x'_1, \ldots, x'_n\}$ ($K'_{ij} = k(x'_i, x'_j)$), this posterior would approach the posterior computed from the "true" prior covariance function K as the number of observations $m \to \infty$. *Thus, GP regression with an informative prior derived from the pretrained embeddings performs the same as GP regression with an uninformative prior derived from random embeddings, as the number of observed samples approaches infinity.*

C Study of Linguistic Properties

We now describe in more detail how we define our metrics for the three linguistic properties for both NER and sentiment analysis tasks (Appendix C.1), as well as provide extended empirical results from our linguistic studies (Appendix C.2).

C.1 Linguistic Properties: Detailed Definitions

We define the metrics in detail below for our three linguistic properties: complexity of text structure (Appendix C.1.1), ambiguity in word usage (Appendix C.1.2), and prevalence of unseen words (Appendix C.1.3). To provide further intuition for these metrics, in Figure 4 we present actual examples from the CoNLL-2003 NER task and the CR sentiment analysis task for each of the metrics, along with the errors made by each embedding type on these examples.

C.1.1 Complexity of Text Structure

We define the following metrics for NER and sentiment analysis to measure the structural complexity of an entity or sentence, respectively:

NER: For NER, we measure the linguistic complexity of an entity in terms of the number of tokens in the entity (e.g., "George Washington" spans 2 tokens), as correctly labeling a longer entity requires understanding the *relationships* between the different tokens in the entity name.

Sentiment analysis: For sentiment analysis, we need a sentence-level proxy for structural complexity; toward this end, we leverage the dependency parse tree for each sentence in the dataset.[10] In particular, we characterize a sentence as more structurally complex if the average distance between dependent words is higher. We consider this definition because long-range dependencies generally require more contextual information to understand. To avoid diluting the average dependency length, we do not include dependencies where either the head or the tail of the dependency is a punctuation or a stop word.

As an example, consider the sentence "George Washington, who was the first president of the United States, was born in 1732". In this sentence, there is a dependence between "George" and "born" of length 14, because there are 13 intervening words or punctuations. This is a relatively large gap between dependent words, and would increase the average dependency length the sentence.

C.1.2 Ambiguity in Word Usage

The next linguistic property we consider is the degree of ambiguity in word usage within a task. To measure the degree of ambiguity in the language, we define the following metrics in the context of NER and sentiment analysis:

NER: For NER as a word-level classification task, we consider the number of labels (person, location, organization, miscellaneous, other) a token appeared with in the training set as a measure of its ambiguity (e.g., "Washington" appears as a person, location, and organization in the CoNLL-2003 training set). For each token in the validation set, we enumerate the number of tags it appears with in the training set.

Sentiment analysis: For sentiment analysis, we measure the ambiguity of a sentence by considering whether the words in the sentence generally appear in positive or negative sentences in the training data. For the binary case, we take the average over words in the sentence of the unigram probability that a word is positive, and then compute the entropy of a coin flip with this probability of being "heads". More specifically, to compute the unigram probability $p(+1 \mid w)$ for a word w, we measure the fraction of training sentences containing w which are positive. Our ambiguity metric is then defined for a sentence S as

$$H\left(\frac{1}{|S|}\sum_{w \in S} p(+1 \mid w)\right),$$

where $H(p) = -p\log_2(p) - (1-p)\log_2(1-p)$ is the entropy of a coin flip with probability p. Intuitively, sentences with generally positive (or negative) words will have low entropy, and be easy to classify even with non-contextual embeddings.

[10] We use the StanfordNLP dependency parser for our metric: `https://pypi.org/project/stanfordnlp/`.

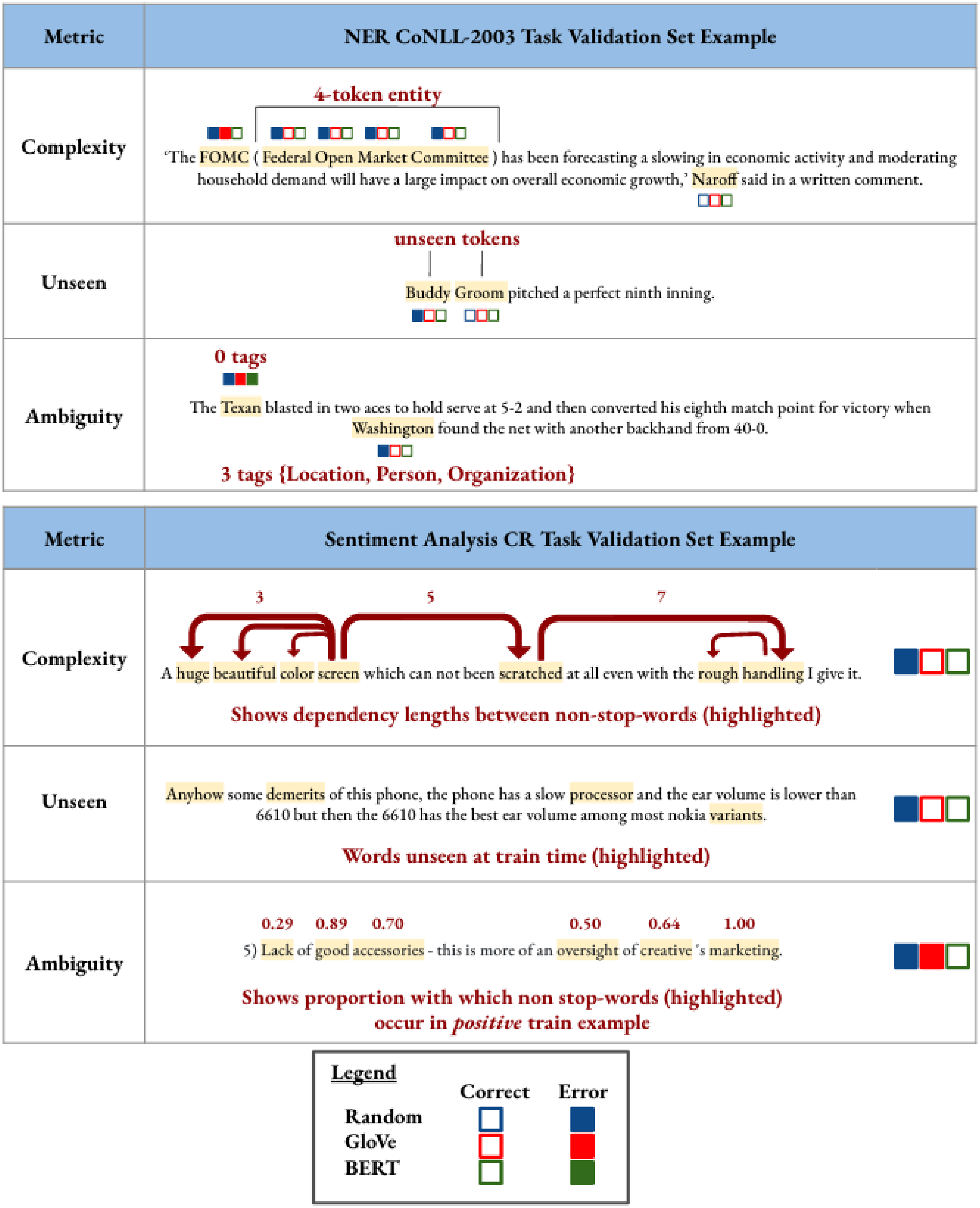

Figure 4: Examples from the CoNLL-2003 NER task (above) and the CR sentiment analysis task (below) validation sets, to provide further intuition for the three linguistic properties. All of the examples above fall in the validation set slices that have metric values above the median, and are thus considered relatively difficult examples according to these linguistic metrics. For example, in the case of NER, (1) the "Federal Open Market Committee" is a relatively long, 4-token entity, (2) "Buddy" and "Groom" are both tokens that were not seen during training, and (3) "Washington" was seen in the training set with three different entity type labels (location, person, organization). In the case of the sentiment analysis examples, (1) the complexity metric sentence has several long dependences (lengths 3, 5, and 7) because it has numerous adjective, adverb, and noun modifiers, (2) the unseen metric sentence has four words that were not seen during training ("anyhow", "demerits", "processor", "variants"), and (3) the ambiguity metric sentence has words that were mainly positive during training ("good", "creative"), as well as words which were mainly negative during training ("lack"). We use empty vs. filled-in squares of different colors to show whether a given embedding type got an example correct vs. incorrect, respectively (see legend).

Category	BERT	Random	GloVe
LS	0.19	0.14	0.13
PAS	0.33	0.20	0.20
L	0.12	0.15	0.13
KCS	0.10	0.17	0.13
Overall	0.500	0.475	0.465

Table 3: The performance (Matthews correlation coefficients) of BERT, random, and GloVe embeddings across the four linguistic categories defined by the GLUE diagnostic task: lexical semantics (LS), predicate-argument structure (PAS), logic (L), and knowledge and common sense (KCS). We also include the overall diagnostic performance.

For non-binary sentiment tasks with C-labels (e.g., $C = 6$ for the TREC dataset), we consider the entropy of the average label distribution $\frac{1}{|S|}\sum_{w \in S} p(y \mid w) \in \mathbb{R}^C$ over the words in the sentence. Here, $p(y \mid w)$ is defined as the fraction of the sentences in the training set containing the word w which had the label y. Note that for stop words and punctuation, we always consider $p(y \mid w)$ as the uniform distribution over the set of possible labels y (for both binary and non-binary classification tasks).

C.1.3 Prevalence of Unseen Words

We define the following metrics for the prevalence of unseen words for NER and sentiment analysis tasks:

NER: For a word in the NER validation set, we consider as our metric the inverse of the number of times the word appeared in the training data (letting $1/0 := \infty$). We consider the *inverse* of the number of training set appearances because intuitively, if a word appears *fewer* times in the training set, we expect it to be *harder* to correctly classify this word at test time—especially for non-contextual or random embeddings.

Sentiment analysis: For sentiment analysis, given a sentence, we consider as our metric the fraction of words in the sentence that were *never* seen during training. More specifically, we count the number of unseen words (that are not stop words), and divide by the total number of words in the sentence. Intuitively, sentences with many unseen words will attain high values for this metric, and will be difficult to classify correctly without prior (i.e., pretrained) knowledge about these unseen words.

C.2 Extended Results

We present the detailed results from our evaluation of the different embedding types on the GLUE diagnostic dataset (Appendix C.2.1), and extended validation of the linguistic properties we define in Section 3.3 (Appendix C.2.2).

C.2.1 GLUE Diagnostic Results

The GLUE diagnostic task facilitates a fine-grained analysis of a model's strengths and weaknesses in terms of how well the model handles different linguistic phenomena. The task consists of 550 sentence pairs which are classified as entailment, contradiction, or neutral. The GLUE team curated the sentence pairs to represent over 20 linguistic phenomena, which are grouped in four top-level categories: lexical semantics (LS), predicate-argument structure (PAS), logic (L), and knowledge and common sense (KCS). We follow the standard procedure and use the model trained on the MNLI dataset (using the random, GloVe, or BERT embeddings) to evaluate performance on the diagnostic task. We report the Matthews correlation coefficient (MCC) performance of the different embedding types on the four top-level categories in Table 3.

Our two key observations are: (1) the non-contextual embeddings (random and GloVe) perform similarly to one another across all four top-level categories; (2) the performance difference between contextual and non-contextual embeddings is most stark for the predicate-argument (PAS) category, which includes phenomena that require understanding the interactions between the different subphrases in a sentence. Within PAS, the BERT embeddings attain a 10+ point improvement in MCC over random embeddings for sentences reflecting the following phenomena: Relative Clauses/Restrictivity, Datives, Nominalization, Core Arguments, Core Arguments/Anaphora/Coreference, and Prepositional Phrases.

C.2.2 GloVe vs. BERT Results

In Table 4, we replicate the results from Table 2, but instead of comparing BERT embeddings to random embeddings, we compare them to GloVe embeddings. We can see that for 11 out of 14 cases for the complexity and ambiguity metrics, the gap between contextual (BERT) and non-contextual (GloVe) performance is larger for the validation slices above the median than below; this aligns with our results comparing random and BERT embeddings.

	Complexity		Ambiguity		Unseen	
Task	Abs.	Rel.	Abs.	Rel.	Abs.	Rel.
NER (CoNLL)	+6.7	4.0	+5.9	3.3	-1.4	0.8
Sent. (MR)	-0.6	0.9	+6.5	2.5	-1.0	0.9
Sent. (SUBJ)	-1.8	0.6	+4.4	6.0	-1.3	0.6
Sent. (CR)	+1.2	1.5	-2.4	0.4	0.0	1.0
Sent. (SST)	+7.8	5.3	+6.0	3.2	-2.8	0.6
Sent. (TREC)	+2.2	1.4	+8.1	4.1	+3.7	1.8
Sent. (MPQA)	+6.6	-3.2	+2.9	-1.8	+0.4	3.0

Table 4: For our complexity, ambiguity, and unseen prevalence metrics, we slice the validation set using the median metric value, and compute the average error rates for GloVe and BERT on each slice. We show that the gap between GloVe and BERT errors is larger above than below the median in 11 out of 14 of the complexity and ambiguity results both in absolute (Abs.) and relative (Rel.) terms; however, on the unseen metrics, this only holds for 2 out of 7 cases, which suggests that GloVe embeddings are able to relatively effectively deal with unseen words.

Interestingly, this is only the case for 2 out of 7 of the cases for the unseen metrics. This is likely because both GloVe and BERT embeddings are able to leverage pretrained semantic information about unseen words to make accurate predictions for them, and thus perform relatively similarly to one another on unseen words.