Contextual Embeddings: When are they worth it?

Simran Arora, Avner May, Jian Zhang, Christopher Ré
Stanford University
{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% or 10% accuracy on benchmark tasks. Furthermore, we identify properties of data for which contextual embeddings give particularly large gains: sentences 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 pre-training, 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

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% or 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?
- **Ambiguity in word usage**: Are words likely to

---

¹This aligns with recent observations from experiments with classic word embeddings at Apple (Ré et al., 2020).
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. on 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 fixed 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., in Limospatam and Collier (2016)) as a simple and efficient non-pretrained baseline. 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., 2018) as a simple and efficient approach (for more details, see Appendix A.1).

2.1 System Efficiency of Embeddings

We discuss the computational and memory requirement of the different embedding methods, focusing on downstream task training and inference.

**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 BERT$_{BASE}$ 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 $n = 400k$).

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 setting pretrained contextual embeddings give large improvements, we compare 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.

---

2Note that one could also simply store the random seed, though this requires regenerating the embedding matrix every time it is accessed.

3Pretrained contextual and non-contextual embeddings also require significant computational resources during pretraining. For example training BERT$_{BASE}$ takes 4 days on 16 TPU chips.
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 BERT\textsubscript{BASE} word embeddings, 300-dimensional publicly available GloVe embeddings, and 800-dimensional random circulant embeddings. We keep the embeddings fixed during training for all embedding types, 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 we use, please see Appendix A.

### 3.2 Impact of Training Data Volume

We show that the amount of 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 increases (plots for all tasks in Appendix C). As a result of this improvement, we show in Table 1 that the performance of the non-contextual embeddings trained on 1x-16x less data.

![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.](image)

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). 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 trained on 1x-16x less data.

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 linguistic categories; we observe that the 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.

### 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.

Specifically, in this table we show for each task the difference between the accuracies attained by BERT vs. GloVe and random (note that for a few tasks random outperforms GloVe!), as well as the
vation 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 contextual embeddings providing large boosts in performance.

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 (See App. A.5 for detailed definitions).

#### Complexity of text structure

We hypothesize that language with more complex internal structure will be harder for non-contextual embeddings.

- **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.

- **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 and location in CoNLL-2003).

- **Sentiment analysis**: As a measure of the ambiguity of a sentence, 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.\(^4\)

#### 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.

- **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 in two, with points with metric values below the median in one half, and above the median in the other. We see that in 19/21 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.

### 4 Conclusion

We compared the performance of contextual embeddings with non-contextual pretrained embeddings and a novel embedding baseline—random word embeddings. We showed that these simpler embeddings perform surprisingly well relative to the contextual embeddings on tasks with plentiful

\(^4\)We exclude stop words in our sentiment analysis metrics.
labeled data and simple language. We hope this work inspires future research on better understanding the differences between embedding methods, and on designing simpler & 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, 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, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In COLING.


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


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


A Experimental Details

A.1 Embeddings

We evaluate random embeddings in comparison to BERT contextual embeddings and GloVe non-contextual embeddings. We specifically used 800-dimensional random embeddings, 768-dimensional BERTBASE, 300-dimensional GloVe embeddings. We freeze each set of embeddings prior to training, and do not fine-tune the embeddings during training. The random and GloVe embeddings are normalized to the same norm.

**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., 2018). 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:

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

This circulant embedding technique has been used in the literature of kernel methods (Yu et al., 2015). As the random embedding matrix is randomly generated, it does not require expensive pretraining over large language corpora. 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.

A.2 Tasks

We present evaluations on three types of standard downstream NLP tasks: named entity recognition (NER), sentiment analysis, and language understanding. NER involves both classifying each token in the input text an entity or 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. We evaluate on five binary sentiment analysis benchmark datasets including MR, MPQA, CR, SST, and SUBJ. We also include evaluation on the benchmark TREC dataset, which assigns one of six labels to each input example. For language understanding, we use the standard GLUE benchmark tasks, and GLUE diagnostic task.

A.3 Downstream Task Models

For NER, we use a BiLSTM task model with a CRF decoding layer and default hyperparameters from the flair (Akbik et al., 2019) repository implementation (https://github.com/zalandoresearch/flair): 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. For 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 learning rate decay constant of 0. We report the validation error rates in evaluations of each task. For the GLUE tasks, 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 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.

A.4 Training Data Experiments

For each task, we evaluate performance using five fractions of the full training dataset, to understand how the amount of training data effects performance: $\{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{3}{4}, 1\}$. For each fraction $c$, we randomly select a subset of the training set of the corresponding fraction, and replicate this data $1/c$ times; we then train models using this redundant dataset, using the model architectures described in Appendix A.3. In downstream training we perform a separate hyperparameter sweep of the learning rate at each fraction of the training data, and se-
lect the best learning rate for each embedding type at each learning rate, to minimize the impact of the learning rate on our experiments. We select a task-specific range of learning rates:

A.4.1 Learning Rates

In downstream training we perform a separate hyperparameter sweep of the learning rate at each fraction of the training data, and select the best learning rate for each embedding type at each fraction based on validation set accuracy, to minimize the impact of the learning rate on our experiments. We sweep the learning rates using the following sets for the different task types.

- **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\}.

A.5 Linguistic Property Experiments

Given the high cost of pretraining and motivated by our observation that the relative performance between contextual and non-contextual embeddings is highly variable, we want to understand the linguistic properties that help explain this difference. We show that high structural complexity, word ambiguity, and prevalence of unseen words correlate with boosted performance for contextual embeddings.

- **Structural complexity** requires learning how multiple words and sub-phrases interact in order to understand the meaning of the sentence as a whole. We hypothesize that language with more complex internal structure will be harder for non-contextual embeddings.

- **We hypothesize** that non-contextual embeddings will perform poorly at disambiguating words that are used in multiple different ways in the training set, as this requires using the specific context of a word to understand which meaning is intended.

- **The prevalence of unseen words** relates to whether the language used at train-time, overlaps with the language used at inference-time. We hypothesize that the performance boost of contextual embeddings will be greater when there are many unseen words in the evaluation data.

These three properties has task specific manifestations since NER is a word level classification task, while sentiment analysis assigns labels at the sentence-level. We give the detailed metrics and implementation details for the three properties below.

A.5.1 Complexity of Text Structure

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

For **NER** as a word-level task, we defined structural complexity in terms of the number of tokens in an 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. Splitting the entities in the validation set by the number of tokens, we calculate the median number of tokens per entity, and split the entities into two groups at the median point. We calculate and report the validation set error rates for each embedding type in the groups below (≤) and above (> ) this median value.

For sentiment analysis, we need a sentence-level proxy for structural complexity and thus leverage the dependency parse tree for each sentence in the dataset. In particular, we characterize a sentence as more structurally complex if the average separation between dependent words is higher. To provide an example of this metric calculation:

For each word \(w\), the dependency parser will output it’s governing word \(w'\), or the word in the sentence that \(w\) depends on. If we consider the sentence: “George Washington, who was the first president of the United States, was born in 1732”, there is a dependence between \(w\) ”George” and \(w'\ ”born”\”. If we index the words in the sentence, there are 14 words separating \(w\) and \(w'\) is. For each dependence \((w, w')\) in the dependency output, we sum this separation between \(w\) and \(w'\). The average separation gives us a measure of the sentence’s structural complexity: long-dependencies generally require more contextual information to understand, due to the separating-words that can modify \(w\) and \(w'\).

For the dependency parser, we use the StanfordNLP Python library (https://pypi.org/project/stanfordnlp/). In the metric calculation, we use standard approaches to minimize the effects of confounding factors: we ignore dependencies for which either \(w\) or \(w'\) is a
punctuation or stop word. For our presented results, we calculate the median average-dependency-separation across sentences in the dataset, and present error rates for the group of sentences both below ($\leq$) and above ($>$) this median value.

A.5.2 Word Ambiguity

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:

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 and location in CoNLL-2003). For each token in the validation set that is also present in the training dataset, we enumerate the number of tags it appears with in the training dataset. We find the median tag-count, and report the validation error rates for each embedding type in the groups below ($\leq$) and above ($>$) the median.

For sentiment analysis, as a sentence-level measure of the ambiguity, we consider whether each word in the sentence generally appears in positive, negative, or mixed sentences in the training data (or the distribution across the six TREC classes in the non-binary case). 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”. For sentence $S$, we compute $p(+1|w)$ for each word $w \in S$:

$$\text{metric} = H\left(\frac{1}{|S|} \sum_{w \in S} p(+1|w)\right)$$

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 $\frac{1}{n} \sum_{i=1}^{n} p(y|x_i) \in \mathbb{R}^6$ over the words $x_i$ the sentence.

Intuitively, sentences with all positive (or negative) words will have low-entropy, and be easy to classify. We calculate the median entropy and show results for the error rates of sentences ($\leq$) and above ($>$) the median. We ignore stop words in this calculation.

A.5.3 Prevalence of Unseen Words

In the context of NER and sentiment analysis, we define the following metrics for the prevalence of unseen words.

For a token in a NER validation set, we simply consider the number of times the token appeared in the training data. Fewer appearances is expected to be more difficult. We compute the median number of appearances and in our results we show the error rate for words that appear more than the median number of times in the ”below” result, and less than the median number of times in the ”above” result.

For sentiment analysis, given a sentence, we consider as our metric the fraction of words in the sentence that were never seen during training. We compute the median fraction and show results for the error rates of sentences ($\leq$) and above ($>$) the median. We ignore stop words in this calculation.

A.6 Textual Examples of Linguistic Properties

In Figure 2, we present actual examples from the NER CoNLL-2003 task, which help the reader further develop intuition for the three linguistic properties we consider in this paper.

B Extended Results: Linguistic Properties

We present the detailed results from our evaluation of the different embedding types on the GLUE diagnostic dataset (Appendix B.1), and extended validation of the linguistic properties we define in Section 3.3 (Appendix B.2).

B.1 GLUE Diagnostic Results

The GLUE diagnostic task facilitates a fine-grained analysis of a model’s strengths and weaknesses in capturing a set of defined linguistic properties. The task consists of 550 sentence pairs which are classified as entailment, contradiction, or neutral. The GLUE team curated and audited 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 word-knowledge (K). We follow the standard procedure and use the MNLI predictor of each model (trained with BERT, GloVe, and random embeddings) to evaluate the diagnostic task. We report the categorical results (MCC) in Table 4.
The FOMC (Federal Open Market Committee) has been forecasting a slowing in economic activity and moderating household demand will have a large impact on overall economic growth, "Naroff said in a written comment.

"FOMC" (S-ORG), "Federal" (B-ORG), "Open" (I-ORG), "Market" (I-ORG), "Committee" (E-ORG) entities, respectively.

"Federal Open Market Committee" = 4-tokens

BERT 'FOMC': (Predicted: 'O')
'Federal': (Predicted: 'B-MISC')
'Open': (Predicted: 'E-MISC')
'Market': (Predicted: 'O')
'Committee': (Predicted: 'O')

GloVe 'FOMC': (Predicted: 'O')

Unseen classifying words that never appear in the training data

Buddy Groom pitched a perfect ninth inning.

Buddy ('B-PER') and Groom('E-PER') never appear in the training data

BERT 'Buddy': (Predicted: 'B-ORG')
'Washington',  (Predicted: 'E-LOC')

GloVe 'Buddy': (Predicted: 'O')
'Washington',  (Predicted: 'E-LOC')

Figure 2: Example sentences from the CoNLL-2003 NER task, to provide further intuition for the three linguistic properties.

B.2 Extended Validation of Linguistic Properties: GloVe vs. BERT

In Table 3, we replicate the results from Table 2, but instead of comparing BERT embeddings to random embeddings, we compare them to BERT. We can see that once again, for most of the tasks, the gap between contextual (BERT) and non-contextual (GloVe) performance is large for the validation slices above the median than below. Interestingly, this is often not the case for the unseen metrics; intuitively, this makes sense, because GloVe embeddings are able to leverage semantic information of unseen words to make accurate predictions.

Our two key observations are: (1) that the GloVe and random embeddings generally perform on par, which motivates our study across the contextual versus non-contextual categories of embeddings; (2) the performance difference between contextual and non-contextual embeddings is most stark for PAS, which includes phenomena that require understanding the interactions between parts of speech or subphrases present in a sentence. Within PAS, the BERT resulted in a 10+ point higher MCC over random for sentences reflecting the following phenomena: Relative Clauses/Restrictivity, Datives, Nominalization, Core Arguments, Core Arguments/Anaphora/Coreference, and Prepositional Phrases.

<table>
<thead>
<tr>
<th>Task</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER (CoNLL)</td>
<td>+6.7</td>
</tr>
<tr>
<td>Sent. (MR)</td>
<td>-0.6</td>
</tr>
<tr>
<td>Sent. (SUBJ)</td>
<td>-1.8</td>
</tr>
<tr>
<td>Sent. (CR)</td>
<td>+1.2</td>
</tr>
<tr>
<td>Sent. (SST)</td>
<td>+7.8</td>
</tr>
<tr>
<td>Sent. (TREC)</td>
<td>+2.2</td>
</tr>
<tr>
<td>Sent. (MPQA)</td>
<td>+6.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Ambiguity</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER (CoNLL)</td>
<td>+5.9</td>
</tr>
<tr>
<td>Sent. (MR)</td>
<td>+6.5</td>
</tr>
<tr>
<td>Sent. (SUBJ)</td>
<td>+4.4</td>
</tr>
<tr>
<td>Sent. (CR)</td>
<td>-2.4</td>
</tr>
<tr>
<td>Sent. (SST)</td>
<td>+6.0</td>
</tr>
<tr>
<td>Sent. (TREC)</td>
<td>+8.1</td>
</tr>
<tr>
<td>Sent. (MPQA)</td>
<td>+2.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER (CoNLL)</td>
<td>-1.4</td>
</tr>
<tr>
<td>Sent. (MR)</td>
<td>-1.0</td>
</tr>
<tr>
<td>Sent. (SUBJ)</td>
<td>-1.3</td>
</tr>
<tr>
<td>Sent. (CR)</td>
<td>0.0</td>
</tr>
<tr>
<td>Sent. (SST)</td>
<td>-2.8</td>
</tr>
<tr>
<td>Sent. (TREC)</td>
<td>+3.7</td>
</tr>
<tr>
<td>Sent. (MPQA)</td>
<td>+0.4</td>
</tr>
</tbody>
</table>

Table 3: 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 12 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.
### Table 4: The performance (Matthews) 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 (K). We also include the overall diagnostic performance.

<table>
<thead>
<tr>
<th>Category</th>
<th>BERT</th>
<th>Random</th>
<th>GloVe</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS</td>
<td>0.19</td>
<td>0.14</td>
<td>0.13</td>
</tr>
<tr>
<td>PAS</td>
<td>0.33</td>
<td>0.20</td>
<td>0.20</td>
</tr>
<tr>
<td>L</td>
<td>0.12</td>
<td>0.15</td>
<td>0.13</td>
</tr>
<tr>
<td>K</td>
<td>0.10</td>
<td>0.17</td>
<td>0.13</td>
</tr>
<tr>
<td>Overall</td>
<td>0.500</td>
<td>0.475</td>
<td>0.465</td>
</tr>
</tbody>
</table>

### C Extended Results: Impact of Training Data Volume

In Figures 3, 4, and 5, 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.

![Figure 3: Performance of random, GloVe, and BERT embeddings on the CoNLL-2003 NER task as we vary the amount of training data](image-url)
Figure 4: Performance of random, GloVe, and BERT embeddings on the sentiment analysis tasks as we vary the amount of training data.
Figure 5: Performance of random, GloVe, and BERT embeddings on GLUE tasks as we vary the amount of training data