BERTering RAMS: What and How Much does BERT Already Know About Event Arguments? — A Study on the RAMS Dataset

Varun Gangal, Eduard Hovy
Language Technologies Institute
Carnegie Mellon University
{vgangal,hovy}@cs.cmu.edu

Abstract

Using the attention map based probing framework from (Clark et al., 2019), we observe that, on the RAMS dataset (Ebner et al., 2020)[1], BERT's attention heads[2] have modest but well above-chance ability to spot event arguments sans *any training or domain finetuning*, varying from a low of 17.77% for *Place* to a high of 51.61% for *Artifact*. Next, we find that linear combinations of these heads, estimated with ≈11% of available total event argument detection supervision, can push performance well-higher for some roles — highest two being *Victim* (68.29% Accuracy) and *Artifact* (58.82% Accuracy). Furthermore, we investigate how well our methods do for cross-sentence event arguments. We propose a procedure to isolate "best heads" for cross-sentence argument detection separately of those for intra-sentence arguments. The heads thus estimated have superior cross-sentence performance compared to their jointly estimated equivalents, albeit only under the unrealistic assumption that we already know the argument is present in another sentence. Lastly, we seek to isolate to what extent our numbers stem from lexical frequency based associations between gold arguments and roles. We propose NONCE, a scheme to create adversarial test examples by replacing gold arguments with randomly generated "nonce" words. We find that learnt linear combinations are robust to NONCE, though individual best heads can be more sensitive.

1 Introduction

The NLP representation paradigm has undergone a drastic change in this decade — moving from linguistic/task motivated 0-1 feature families to per-word-type pretrained vectors (Pennington et al., 2014) to contextual embeddings (Peters et al., 2018).

Contextual embeddings (CEs) produce in-context representations for each token - the representation framework being a large, pretrained encoder with per-token outputs. The typical procedure to use CEs for a downstream task is to add one or more task layers atop each token, or for a designated token per-sentence, depending on the nature of the task.

The task layers (and optionally, the representation) are then "*finetuned*" using a task specific loss, albeit with a slower training rate than would be used for from-scratch training. ELMo (Peters et al., 2018) was an early CE. The three-fold recipe of a transformer based architecture, masked language modelling objective and large pre-training corpora, starting with BERT (Devlin et al., 2018) led to CEs which were vastly effective for most tasks.

The strong performance of contextual representations with just shallow task layers and minimal finetuning drove the urge to understand what and how much these models *already knew* about aspects of syntax and semantics. The study of methods and analysis to do this has come to be called *probing*. Besides "explaining" CE featurization, probing can aid in finding lacunae to be addressed by future representations.

Linzen et al. (2016), one of the early works on probing, evaluated whether language models could predict the correct verb form agreeing with the noun. Marvin and Linzen (2018) generalized this approach beyond single-word gaps with a larger suite of "minimal pairs". They also control for lexical confounding and expand the probing to new aspects such as reflexive anaphora and NPIs. Gulordava et al. (2018) evaluate subject-verb agreement but only through "nonce" sentences to con-

[1]Refer to Figure 1 of that paper for an example illustrating four role names. Since these role names are human readable and intuitively named, we refer to them without elaboration.

[2]We use *map* to refer to the per-example word-word activations at a particular layer-head, while *head* refers either to the identity of the particular layer-head. We ground these terms more clearly in §2.1

trol for both lexical confounding and memorization[3]. Lakretz et al. (2019) isolate units of LSTM language models whose activations closely track verb-noun number agreement, particularly for hard, long-distance cases. Clark et al. (2019), whose probing methods we adopt, examine if BERT attention heads capture dependency structure.

In this work, we probe what and how much a pre-trained BERT representation already knows about event roles and their arguments. Understanding how well event arguments are represented can be a first foray into understanding other aspects about events. Extraction of event arguments is often a prerequisite for more complex event tasks. Some examples are event coreference (Lu and Ng, 2018), detecting event-event temporal (Vashishtha et al., 2019) and causal relations (Dunietz et al., 2017), sub-event structure (Araki et al., 2014) and generating approximate causal paths (Kang et al., 2017). Tuples of event-type and arguments are one way of inducing script like-structures (Chambers and Jurafsky, 2008). In summary, our work makes the following contributions:

1. We show that there always exists a BERT attention head (BESTHEAD) with above-chance ability to detect arguments, for a given event role. We also show that this ability is even stronger through learnt linear combinations (LINEAR) of heads.
2. We notice a relative weakness at detecting cross sentence arguments (§3.3). Motivated by this, we devise a procedure to isolate only the cross-sentence argument detection ability of heads w.r.t a role (§3.3.1). Our procedure considerably improves cross-sentence performance for some roles (§3.4), especially for INSTRUMENT and PLACE.
3. Lastly, we seek to isolate how much of the zero-shot argument detection ability originates solely from the model's world knowledge and lexical frequency based associations. To do this, we propose NONCE, a method to perturb test examples to dampen such associations (§2.5.3). We find that the LINEAR approach is robust to NONCE perturbation, while BESTHEAD is more sensitive.

[3]A motivation for our ablation in §2.5.3

2 Methodology

2.1 Background

2.1.1 Transformers

The Transformer architecture (Vaswani et al., 2017) consists of $|L|$ layers, each comprised of $|H| > 1$ "self-attention" heads. Here, we describe the architecture just enough to ground terminology - we defer to the original work for detailed exposition.

In a given layer l[4], a single self-attention head h consists of three steps - First, query, key and value projections $q_i^h = {Q_h}^T e_i, k_i^h = {K_h}^T e_i, v_i^h = {V_h}^T e_i$ are computed from the previous layer's token embedding e_i. Then, softmax normalized dot products $\alpha_{ij}^h = \frac{(q_i^h)^T {k_j}^h}{\sum_m (q_i^h)^T {k_m}^h}$ are computed between the current token's query projection and other token's key projections. These dot products a.k.a *attention values* are then used as weights to combine all token value projections - $o_i^h = \sum_j \alpha_{ij}^h v_j^h$ gives the current head's token output o_i^h. Finally, the outputs from all heads are concatenated and projected to get the per-token embeddings for the current layer $o_i = W^T Concat(\{o_i^0, o_i^1 \ldots o_i^{|H|-1}\})$

Henceforth, we refer to the parameter tuple $\{Q_{h,l}, K_{h,l}, V_{h,l}\}$, uniquely identified by $h \in \{0, 1, \ldots |H| - 1\}, l \in \{0, 1, \ldots |L| - 1\}$ as the "*attention head*" or simply "*head*", while values $\alpha_{ij}^{h,l}$ are collectively referred to as the "*attention map*".

2.1.2 BERT

BERT uses a Transformer architecture with 12 heads and 12 layers[5]. It comes with an associated BPE tokenizer (Sennrich et al., 2015) which tokenizes raw inputs to subwords. Its vocabulary contains three special tokens - [CLS], [SEP] and [MASK]. While [CLS] and [SEP] serve as start and end (or sequence-separator) tokens, [MASK] is used in pretraining as described next.

BERT follows a two-stage pretraining process. In the first stage, also known as masked language modelling (MLM), randomly selected token positions are replaced with [MASK]. The task is to predict the true identities of words at these positions, given the sequence. This stage uses single sentences as training examples. In the second stage, also known as next sentence prediction (NSP), the

[4]We omit layer index l in the rest of the passage to declutter our notation.

[5]For *bert-base*. *bert-large* uses 24 heads and 24 layers.

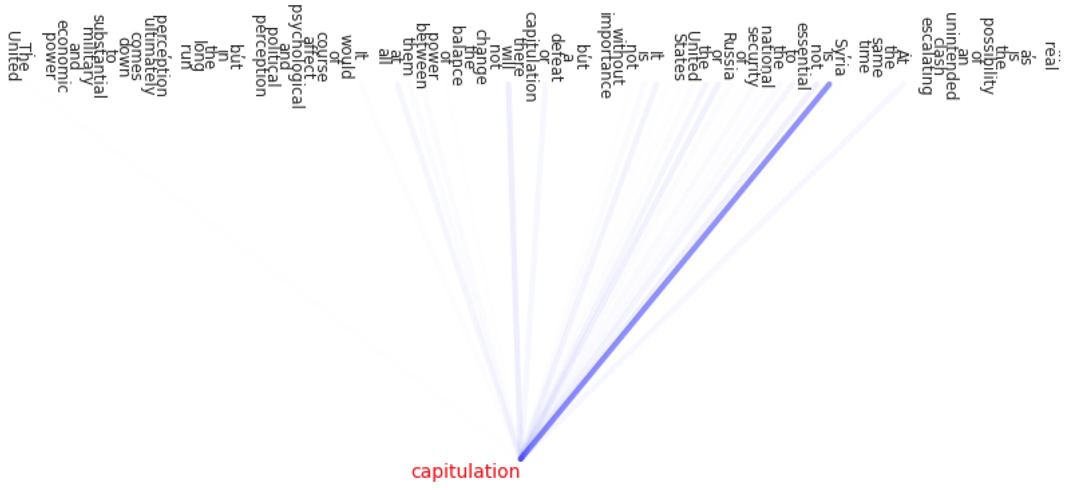


Figure 1: In this example, the head chosen by BESTHEAD for the PLACE role, correctly picks out the argument "*Syria*" for the trigger "*capitulation*". Attention probabilities are shown as blue lines from trigger token to other tokens, with boldness indicating magnitude. It manages to evade distractor pronouns (*there*) and other geographical entity names (*Russia* and *United States*). The text above flows in right to left direction. The full text reads: "*Chances of intentional conflict are real as is the possibility of an unintended clash escalating . At the same time, Syria is not essential to the national security of Russia or the United States. It is not without importance but a defeat or* ***capitulation*** *there will not change the balance of power between them at all . . .*"

model is given a pair of sentences (separated by [SEP]), with the task being to predict whether these were truly consecutive or not.

Unless otherwise mentioned, we use the *bert-base-uncased* model. We use the implementation of BERT from HuggingFace. [6] (Wolf et al., 2019)

2.2 Dataset

We use the recently released RAMS dataset (Ebner et al., 2020) for all our experiments. The reasons for using this particular dataset for our analysis are

- It has a wide mix of reasonably frequent roles (represented well across splits) from different kinds of frames . Discussion on non-frequent roles can be found in §3.6.

- For many roles, it has examples with the gold arguments being in a different sentence from the event trigger. This makes it easy to probe for intra-sentence and cross-sentence argument extraction in the same set of experiments. Analysis of cross-sentence performance can be found in §3.3 and §3.4

We note that the dataset is in English (Bender and Friedman, 2018) and observations made may not generalize to other languages.

2.2.1 Setup

For example x, we refer to the event, role, gold argument and document as e, r, a and D. D is an ordered sequence of tokens $\{w_0, w_1 \dots w_{|D|-1}\}$. i_e denotes the event trigger index[7].

We use the layer index l and head indices 0 to $|H| - 1$ to index the respective head's attention distribution *from* token i to all other tokens $j \in D$ at index i_e.

$$P^*_{l,h}(j|i_e) = \alpha^{i_e j}_{l,h}, 0 \leq l < |L|, 0 \leq h < |H| \quad (1)$$

Note, however that there exist a complementary set of attention values from each token j *to* the token i_e. To use a unified indexing scheme to refer to these values, we use negative indices from -1 to $-|H|$ as their head indices. Since these values come from attention-head activations of different positions, they need to be renormalized to use them as probabilities.

$$P^*_{l,-h}(j|i_e) = \frac{\alpha^{j i_e}_{l,h-1}}{\sum_{k \in D} \alpha^{k i_e}_{l,h-1}}, 0 < h \leq |H| \quad (2)$$

2.2.2 Words and Subwords

Our above framework assumed that the attention maps are between whole word tokens. However, BERT represents a sentence as a sequence of BPE-subwords at every level, including for the attention maps.

We use the quite intuitive approach described in Section 4.1 of (Clark et al., 2019) - incoming attentions to constitutent subwords of a word are added to get the attention to that word. Outgoing attention

[6]github.com/huggingface/transformers/

[7]To simplify our analysis, we do not include multi-word triggers. These form only ≈1.6% of the cases in the dataset.

values from constituent subwords are averaged to get the outgoing attention value from the word.

Note that the above operations precede the probability computations in Equations 1 and 2.

2.2.3 Dataset Splits

We follow the practice of earlier probing works such as (Sorodoc et al., 2020) and (Linzen et al., 2016) of using one of the smaller splits for training. Specifically, we use the original dev split of RAMS (924 examples in total) as our training split. Note that each example could contain multiple role-argument pairs.

Splits	Examples	Tokens
Train (Original Dev)	924	0.12M
Dev (Original Test)	871	0.11M
Test (Original Train)	7329	0.98M

Table 1: Split example counts and token sizes from the RAMS. Note that we use different splits since our work is a probing exercise.

2.3 Evaluation Measure

For a given event e and role r, we define a predicted argument token index $\widehat{a}$ to be accurate if it corresponds to any of the tokens in the gold argument span $[a_{r,e}^{beg}, a_{r,e}^{end}]$. This is described formally in Equation 3. I stands for the 0-1 indicator function.

$$Acc_{e,r,a}(\widehat{a}) = I(a_{r,e}^{beg} \leq \widehat{a} < a_{r,e}^{end}) \qquad (3)$$

Typical measures of argument extraction differ from the one we use, being span-based. Given the limitations of our probing approaches, we lack a clear mechanism of predicting multi-word spans, and can only predict likely single tokens for the argument, which led us to choose this measure[8].

2.4 Approaches

2.4.1 BESTHEAD

Let $X = \{e_m, r_m, a_m\}_{m=1}^{m=M}$ be the training set. X_r is the subset of training examples with $r_m = r$. For each role r, BESTHEAD selects the head $\{l,h\}_{best}(r)$ with best aggregate accuracy on X_r. Other than one pass over the training set for comparing aggregate accuracies of heads for each role, there is no learning required for this method. At test-time, based on the test role, the respective best head is used to predict the argument token.

[8]We will interchangeably refer to Acc as just "*accuracy*" in plain-text in the rest of the paper

$$Acc_{l,h}^{X_r} = \sum_{m=1}^{m=M_r} Acc_{e_m,r,a_m}(\arg\max_j \widehat{P}_{l,h}(j|i_{e_m}))$$

$$\{l,h\}_{best}(r) = \arg\max_{l,h} Acc_{l,h}^{X_r}$$

2.4.2 LINEAR

The LINEAR model learns a weighted linear combination of all $|L| \times |H| \times 2$ head distributions (twice for the "*from*" and "*to*" heads).

$$\phi(j|i) = \sum_{l=0}^{l=|L|-1} \sum_{h=-|H|}^{h=|H|-1} w_{l,h}\widehat{P}_{l,h}(j|i) + B$$

$$\widehat{P}(j|i) = \frac{\phi(j|i)}{\sum_{k=0}^{k=|D|-1} \phi(k|i)}$$

Note that gradients are not backpropagated into BERT - only the linear layer parameters $w_{l,h}, B$ are updated during backpropagation. This formulation is the same as the one in (Clark et al., 2019).

For our loss function, we use the KL Divergence $KL(\widehat{P}||P)$ between the predicted distribution over document tokens $\widehat{P}$ and the gold distribution over document tokens P. For the gold distribution over arg tokens, the probability mass is equally distributed tokens in the argument span, with zero mass on the other tokens.

$$KL(\widehat{P}||P) = \sum_{k=0}^{k=|D|-1} \widehat{P}(k|i) \log \frac{\widehat{P}(k|i)}{P(k|i)}$$

2.5 Baselines

2.5.1 RAND

The expected accuracy of following the strategy of randomly picking any token i from the document D as the argument (other than the trigger word i_e itself). For a given role r with a gold argument $a_{r,e}$ of length $|a_{r,e}|$, this equals $\frac{|a_{r,e}|}{|D|-1}$.

2.5.2 SENTONLY

The expected accuracy of following the strategy of randomly picking any token from the same sentence S_e as the argument, save the event trigger itself. This is motivated by the intuition that event arguments mostly lie in-sentence. This equals $\frac{|a_{r,e}|}{|S_e|-1}$

2.5.3 NONCE procedure

We wish to isolate how much of the heads performance is due to memorized "world knowledge" and typical lexical associations e.g *Russia* would typically always be a PLACE or TARGET. Recent works have shown that BERT does retain such associations, including for first names (Shwartz et al., 2020), and enough so that it can act as a reasonable knowledge base (Petroni et al., 2019).

One way of implementing this is to create perturbed test examples where gold arguments are replaced with synthetically created "nonce" words not necessarily related to the context. This is similar to the approach of (Gulordava et al., 2018).

- Each gold argument token is replaced by a randomly generated token with the same number of characters as the original string.
- Stop words such as determiners, pronouns, and conjunctions are left unaltered, though they might be a part of the argument span.
- We also ensure that the shape of the original argument, i.e the profile of case, digit vs letter is maintained[9]. e.g *Russia-15* can be randomly replaced by *Vanjia-24*, which has the same shape *Xxxx-dd*.
- Note that we do not take pronounceability of the nonce word into account. Though this could arguably be a relevant invariant to maintain, we were not sure of an apt way to enforce it automatically.
- We also note that BERT may end up using a likely larger number of subword tokens to replace the nonce words than it would use for the gold argument token. Since these are essentially randomly composed tokens, they can contain subwords which are rarely seen in vocabulary tokens.

We refer to this procedure as NONCE, and overloading the term, the test set so created as the NONCE test set.

3 Experiments

3.1 Spotting the Best Head

In Table 2, we record the accuracies and layer positions of best heads for the 15 most frequent roles.

1. BESTHEAD always has higher accuracy than the RAND and SENT baselines.
2. 5 of the 15 roles can be identified with 40%+ accuracy - the highest being COMMUNICATOR , at 51.61%.

[9]We are aware that case mostly doesn't matter since we use bert-*-uncased in most experiments

Role	$\{l,h\}_{best}$	%Accuracy
DEFENDANT	8,10	35.90
DESTINATION	0,8	21.43
ORIGIN	7,-1	31.82
TRANSPORTER	8,10	31.58
INSTRUMENT	9,7	31.37
BENEFICIARY	8,10	26.56
ATTACKER	7,-8	33.93
TARGET	9,1	**44.61**
GIVER	8,10	25.55
VICTIM	9,1	**46.34**
ARTIFACT	4,-6	**50.42**
COMMUNICATOR	8,10	**51.61**
PARTICIPANT	8,10	28.57
RECIPIENT	7,10	**40.78**
PLACE	9,1	17.77

Table 2: Best layer-head pair , $\{l,h\}_{best}$ and % Accuracy for the 15 most frequent roles in RAMS, using *bert-base-uncased*. +ve h indices denote "from" heads, while -ve indices denote "to" heads, as explained in §2.2.1

3. The best head for arguments which are not together present in frames is often the same. For instance, Layer 8, Head 10 is the best head for TRANSPORTER, ATTACKER, COMMUNICATOR and BENEFICIARY.
4. Most best heads are located in the higher layers, specifically the 7th, 8th or 9th layers. An exception are the best head for DESTINATION and ARTIFACT roles, located in the 0th layer and 4th layers respectively.
5. *Place* roles are the hardest to identify, with an accuracy of 17.77%.
6. Layer 8, Head 10 seems to be doing a lot of the heavylifting. For 7 out of 15 roles, this is the best head. This shows that it is quite "overworked" in terms of the number of roles it tracks. Furthermore, though some of these role pairs are from different frames (e.g see Point 3 above), some aren't, e.g GIVER and BENEFICIARY. In such cases, atleast one of the two arguments predicted for these two roles is sure to be inaccurate - e.g the head would point to either the GIVER or BENEFICIARY, but not both. [10]
7. Most of the best heads for roles are "from" heads rather than "to" heads, apart from those for ORIGIN, ATTACKER and ARTIFACT.

3.2 LINEAR Performance

Table 3 shows test accuracies for both LINEAR and BESTHEAD approaches, and also the baselines.

For 12 of the 15 roles, LINEAR has higher accuracy than BESTHEAD. There are three exceptions - ORIGIN and INSTRUMENT, which suffer a decline

[10]It is quite non-intuitive for GIVER and BENEFICIARY spans to overlap — we don't see any examples with the same.

Role	RAND	SENTONLY	BESTHEAD	LINEAR
DEFENDANT	1.75	6.98	35.90	56.41
DESTINATION	1.91	7.67	21.43	39.28
ORIGIN	1.36	7.27	31.82	28.79
TRANSPORTER	1.63	6.57	31.58	43.42
INSTRUMENT	1.88	6.29	31.37	25.49
BENEFICIARY	1.34	6.28	26.56	34.37
ATTACKER	2.07	8.52	33.93	46.43
TARGET	1.78	7.30	44.61	44.61
GIVER	1.58	6.29	25.55	32.22
VICTIM	1.50	6.42	46.34	68.29
ARTIFACT	1.86	7.62	50.42	58.82
COMMUNICATOR	1.58	6.55	51.61	63.71
PARTICIPANT	1.49	6.19	28.57	30.72
RECIPIENT	1.83	8.57	40.78	44.69
PLACE	1.67	6.84	17.77	31.93

Table 3: Test accuracies using all the baselines and probe approaches described in §§2.4 for the 15 most frequent roles in RAMS. Both BESTHEAD and LINEAR probes outdo the baselines. LINEAR usually does better, but not for all roles (e.g ORIGIN). Refer to §3.2 for a longer discussion.

and TARGET, which remains the same. A possible reason could be the higher fraction of cross-sentence gold arguments for these roles. The five roles with lowest number of intra-sentence arguments are DESTINATION (58.92%), INSTRUMENT (62.74%), ORIGIN (68.18%), PLACE (70.48%) and TARGET (81.53%).

While DESTINATION and PLACE do see increases in LINEAR compared to BESTHEAD, it could be the case that none of the individual heads are particularly good at capturing cross-sentence arguments for the other three roles, while the best head is already good enough to capture the intra-sentence case. This would make LINEAR not any more rich as a hypothesis space compared to BESTHEAD - causing the similar or slightly worse accuracy. In §3.3 we dig deeper into the aspect of cross-sentence performance.

3.3 Cross-Sentence Performance

From Table 4, we observe that both BESTHEAD and LINEAR performance degrades in the cross-sentence case i.e when "trigger sentence" and "gold argument sentence" differ. Three potential reasons:

1. There are too few instances of cross-sentence event arguments in the small supervised set we use. Furthermore, even if there are a sufficient quantum of cross-sentence event arguments, these form a much smaller proportion of the total instances in comparison to the intra-sentence instances.
2. Because the limited number of attention heads are already dominated by intra-sentence aspects such as dependency relations, punctuation and subject-verb agreement (Clark et al., 2019), it is difficult for a single attention head to have a higher value for outside sentence tokens compared to in-sentence ones.
3. Different heads might be best for intra and cross-sentence performance, and finding one best head for both could be sub-optimal.

Role	BESTHEAD+CSO	LINEAR+CSO
ORIGIN	4.76 (31.82→0.00)	4.76 (56.41→16.34)
INSTRUMENT	47.37 (31.37→21.22)	52.63 (25.49→31.51)
PARTICIPANT	24.99 (28.57→6.37)	29.16 (30.72→6.37)
PLACE	15.31 (17.77→10.18)	30.61 (31.93→9.30)

Table 4: Accuracies on cross-sentence test examples using BESTHEAD+CSO and LINEAR+CSO. The values $Acc_{Total} \rightarrow Acc_{Cross}$ in parentheses are the total test accuracy and cross-sentence test accuracy respectively, using the simple version of the same approach i.e BESTHEAD and LINEAR. The % of cross-sentence examples for each role are: {ORIGIN:31.82 INSTRUMENT:37.26 PARTICIPANT:17.14 PLACE:29.52}

3.3.1 Cross Sentence Occlusion (CSO)

Motivated by the above reasons, we devise a procedure which we refer to as cross-sentence occlusion (CSO). Since Reason 1 is a property of the data distribution, we attempt to alleviate Reasons 2 and 3. To address Reason 3, we try to learn a different head (combination) for the cross-sentence case. To address Reason 2, while finding the best cross-sentence head, we zero-mask out the attention values corresponding to in-sentence[11] tokens and re-normalize the probability distribution.

In practice, one would not be able to use two separate argument detectors for the intra and cross-sentence cases for the same role, since ground-truth information of whether the argument is cross-sentence would be unavailable. We assume this contrived setting only to allow easy analysis[12], and to gloss over the lack of an intuitive zero-shot mechanism of switching between the two cases, when predicting arguments using just attention heads.

3.4 +CSO Results

From Table 4, we can observe the improvement in cross-sentence test accuracy when using the +CSO approach over its simple counterpart, both for BESTHEAD and LINEAR. The only exception to this is the ORIGIN role, where LINEAR betters LINEAR-CSO.

For the INSTRUMENT role, both BESTHEAD+CSO and LINEAR+CSO get close to 50%

[11] RAMS comes with a given sentence segmentation.

[12] And also so that we can validate our diagnosis for poor cross-sentence performance in §3.3

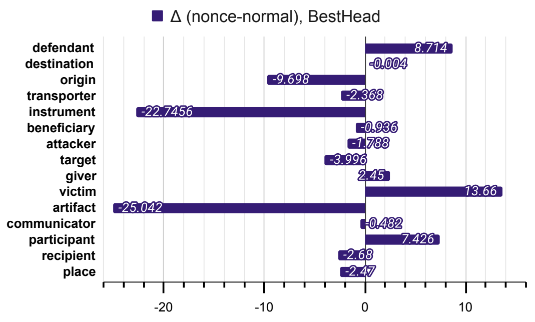
accuracy. In part, their relatively stronger performance can be explained by BESTHEAD and LINEAR already being relatively better at detecting cross-sentence INSTRUMENT (just above 20%, but higher than the sub-15 accuracies on the other roles). Nevertheless, CSO still leads to a doubling of accuracies for both approaches.

We highlight here again that these numbers are only on that subset of the test set where we know that the gold arguments are located in other sentences - though this setting is useful for analysis, a model actually solving this task won't have access to this information.

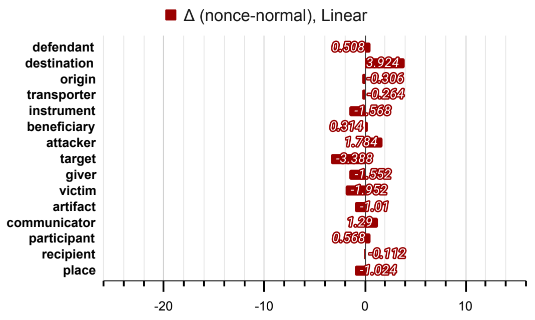
Even in our case, there is no obvious way to have a consolidated probe which uses a LINEAR+CSO and LINEAR component together, since this would require learning an additional component which predicts whether the gold arguments lie intra-sentence or across-sentence.

3.5 Effect of NONCE

In Figures 2a and 2b, we compare the performances of our methods on perturbations of the test set created using the NONCE procedure outlined in §2.5.3 with their normal test performance. Since NONCE is stochaistic, corresponding results are averaged over NONCE sets created with 5 different seeds.



(a) BESTHEAD



(b) LINEAR

Figure 2: Difference in a) BESTHEAD and b) LINEAR accuracies over normal and NONCE test sets

BESTHEAD test performance is more sensitive to NONCE than LINEAR. Especially for INSTRUMENT, ARTIFACT and ORIGIN, the decrease in accuracy is quite drastic. Surprisingly, we also see increases for 4 of the 15 roles - DEFENDANT, GIVER, VICTIM and PARTICIPANT. All other roles see small decreases. For LINEAR, however, most roles are largely unmoved by NONCE, showing that LINEAR relies less on lexical associations.

3.6 Non-Frequent Roles

So far, we've focussed on analyzing the 15 most frequent roles. In this subsection, we also evaluate our approaches for some non-frequent roles outside this set, such as PREVENTER and PROSECUTOR. The results are presented in Table 5. Note that, owing to high sparsity for these roles, these results should be taken with "a pinch of salt" (which is why we chose to separate them out from the frequent roles).

For the frequent roles, we had seen that LINEAR was mostly better than, or equally good as BESTHEAD. For the non-frequent roles, we see that the comparative performance of LINEAR vs BESTHEAD varies a lot more - LINEAR is better for 6 of the 11 roles, and worse for the other 5. The fall in LINEAR performance is largest for PROSECUTOR (58.33 $\rightarrow$ 16.67).

We conjecture that this drop is due to poor generalization as a result of learning from lesser supervision as a result of the roles being non-frequent. Since BESTHEAD has only two parameters (identity of the best head) compared to the 289 parameters of LINEAR, the latter is more sensitive to this problem.

Secondly, we notice that the gap between BESTHEAD and the RAND and SENTONLY baselines is much narrower. For VEHICLE and MONEY, SENTONLY even outdoes BESTHEAD. For VEHICLE, the BESTHEAD accuracy even drops to 0. However, in all these cases, we find that LINEAR still manages to outdo both baselines. We conjecture that these cases could be due to the best head predicted not being very generalizable due to small training set size (for that role). Though LINEAR would also suffer from poor generalization in this case, it might stand its ground better since it relies on multiple heads rather than just one.

3.7 Cased vs Uncased

In our analysis so far, we have been using the same contextual embedding mechanism through-

Role	RAND	SENTONLY	BESTHEAD	LINEAR
PREVENTER	1.51	7.30	15.00	43.14
PASSENGER	1.46	6.79	57.45	45.45
CRIME	3.06	12.62	25.81	57.99
INJURER	1.52	6.51	29.03	16.13
EMPLOYEE	1.47	6.87	53.85	50.00
KILLER	1.75	10.63	14.29	47.62
MONEY	1.51	7.77	4.17	25.00
DETAINEE	1.60	8.39	62.50	50.00
PROSECUTOR	1.55	6.34	58.33	16.67
JUDGECOURT	1.33	6.78	22.22	41.67
VEHICLE	1.47	6.15	0	18.18

Table 5: Test accuracies using all the baselines and probe approaches described in §§2.4 for some non frequent roles. Both BESTHEAD and LINEAR probes still outdo the baselines in most cases, but not as convincingly as for frequent roles. Unlike the frequent roles case, LINEAR actually does worse than BESTHEAD for many roles.

out, namely *bert-base-uncased*. In Figure 3a, we plot the difference of BESTHEAD test accuracies when using *bert-base-cased* vs *bert-base-uncased*.

We can see that *bert-base-uncased* is better for most roles - except for *Attacker*, *Victim* and *Artifact*. We also notice that the best layer-head configuration $\{l_{best}, h_{best}\}$ is mostly not preserved between the *bert-base-cased* and *bert-base-uncased* scenarios. The difference between *bert-base-uncased* and *bert-base-cased* is even more drastic in the cross sentence only experiment , for instance, while there exists a single head which can find cross-sentence *Instrument* args with 37% accuracy, the best such head of *bert-base-cased* has only 17% accuracy.

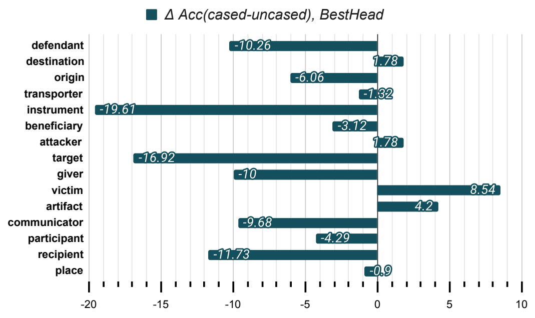
3.8 Qualitative Examples

In Figure 3.8, we illustrate some examples of BESTHEAD identifying arguments. We defer further discussion to Appendix §A owing to lack of space.

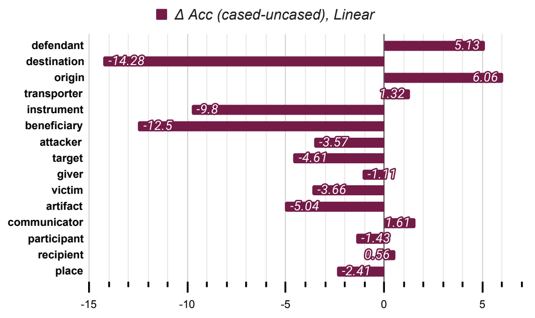
4 Related Work

A complete description of the large body of work on probing is beyond the scope of this paper. Besides those discussed earlier, other aspects studied include filler-gap dependencies (Wilcox et al., 2018), function word comprehension (Kim et al., 2019), sentence-level properties (Adi et al., 2016) and negative polarity items (Warstadt et al., 2019).

Probing is not limited to examining pairwise word relations or sentence properties. Hewitt and Manning (2019) find that BERT token representations are linearly projectable into a space where they embed constituency structure. Recently, Sorodoc et al. (2020) probed transformer based language models for coreference. However, they restrict themselves to entity coreference. Furthermore, they exclude MLMs like BERT from their analysis.



(a) BESTHEAD



(b) LINEAR

Figure 3: Δ in Test accuracy of a) BESTHEAD b) LINEAR using *bert-base-uncased* vs *bert-base-cased*

Hewitt and Liang (2019) raised a note of caution about classifier based probes, pointing out that probes themselves could be rich enough to learn certain phenomena even with random representations. We avoid direct classifier-based probing, thus avoiding the mentioned pitfalls.

5 Conclusion

We showed how BERT's attention heads have modest but well above chance ability to detect arguments for event roles. This ability is achievable either with only i) 2 parameters per role (BESTHEAD) ii) 289 parameters per role (LINEAR). Furthermore, the supervision required to reach this is just $\approx 11\%$ of full training set size. Secondly, we propose a method to learning separate heads (combinations) for cross-sentence argument detection. Our experiments show that the heads so learnt have higher cross-sentence accuracy. Thirdly, we show that LINEAR performance is robust to a perturbed NONCE test setting with weakened lexical associations. In future, we plan to extend our probing to other event aspects like coreference and subevents.

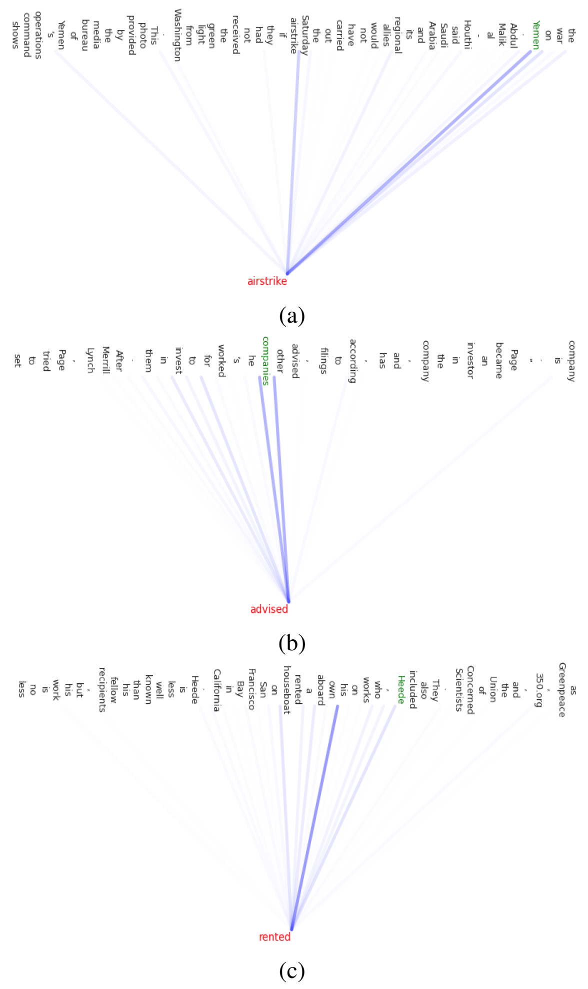


Figure 4: In (a), BESTHEAD correctly picks out the TARGET of "*airstrike*" as "*Yemen*". In (b), BESTHEAD correctly picks out the RECIPIENT of "*advised*" as "*companies*". In (c), the token picked is coreferent but not identical to the gold argument. Attentions are shown as blue lines from trigger token, with lineweight $\propto$ value. Gold arguments are shaded green .

Acknowledgments

We thank Nikita Moghe and Hiroaki Hayashi as well as the three anonymous reviewers for their valuable feedback.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*.

Jun Araki, Zhengzhong Liu, Eduard H Hovy, and Teruko Mitamura. 2014. Detecting Subevent Structure for Event Coreference Resolution. In *LREC*, pages 4553–4558.

Emily M Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does BERT look at? an analysis of BERT's attention. *arXiv preprint arXiv:1906.04341*.

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*.

Jesse Dunietz, Lori Levin, and Jaime G Carbonell. 2017. The BECauSE corpus 2.0: Annotating causality and overlapping relations. In *Proceedings of the 11th Linguistic Annotation Workshop*, pages 95–104.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. Multi-sentence argument linking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8057–8077, Online. Association for Computational Linguistics.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. *arXiv preprint arXiv:1803.11138*.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. *arXiv preprint arXiv:1909.03368*.

John Hewitt and Christopher D Manning. 2019. 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, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Dongyeop Kang, Varun Gangal, Ang Lu, Zheng Chen, and Eduard Hovy. 2017. Detecting and explaining causes from text for a time series event. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2758–2767.

Najoung Kim, Roma Patel, Adam Poliak, Alex Wang, Patrick Xia, R Thomas McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, et al. 2019. Probing what different NLP tasks teach machines about function word comprehension. *arXiv preprint arXiv:1904.11544*.

Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. The emergence of number and syntax units in LSTM language models. *arXiv preprint arXiv:1903.07435*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Jing Lu and Vincent Ng. 2018. Event Coreference Resolution: A Survey of Two Decades of Research. In *IJCAI*, pages 5479–5486.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. *arXiv preprint arXiv:1808.09031*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Vered Shwartz, Rachel Rudinger, and Oyvind Tafjord. 2020. " you are grounded!": Latent Name Artifacts in Pre-trained Language Models. *arXiv preprint arXiv:2004.03012*.

Ionut Sorodoc, Kristina Gulordava, and Gemma Boleda. 2020. Probing for Referential information in Language Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4177–4189.

Siddharth Vashishtha, Benjamin Van Durme, and Aaron Steven White. 2019. Fine-grained temporal relation extraction. *arXiv preprint arXiv:1902.01390*.

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 *Advances in Neural Information Processing Systems*, pages 5998–6008.

Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun Liu, Alicia Parrish, et al. 2019. Investigating Bert's knowledge of Language: Five Analysis Methods with NPIs. *arXiv preprint arXiv:1909.02597*.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do RNN Language models learn about Filler-Gap Dependencies? *arXiv preprint arXiv:1809.00042*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.