A Cross-Task Analysis of Text Span Representations

Shubham Toshniwal, Haoyue Shi, Bowen Shi, Lingyu Gao, Karen Livescu, Kevin Gimpel
Toyota Technological Institute at Chicago
`{shtoshni, freda, bshi, lygao, klivescu, kgimpel}@ttic.edu`

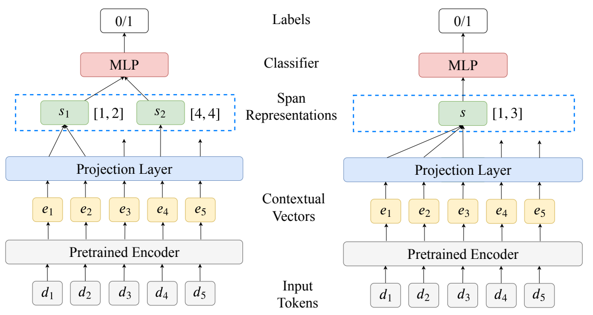
Abstract

Many natural language processing (NLP) tasks involve reasoning with textual spans, including question answering, entity recognition, and coreference resolution. While extensive research has focused on functional architectures for representing words and sentences, there is less work on representing arbitrary spans of text within sentences. In this paper, we conduct a comprehensive empirical evaluation of six span representation methods using eight pretrained language representation models across six tasks, including two tasks that we introduce. We find that, although some simple span representations are fairly reliable across tasks, in general the optimal span representation varies by task, and can also vary within different facets of individual tasks. We also find that the choice of span representation has a bigger impact with a fixed pretrained encoder than with a fine-tuned encoder.

1 Introduction

Fixed-dimensional span representations are often used as a component in recent models for a number of natural language processing (NLP) tasks, such as question answering (Lee et al., 2016; Seo et al., 2019), coreference resolution (Lee et al., 2017), and constituency parsing (Stern et al., 2017; Kitaev and Klein, 2018; Kitaev et al., 2019, *inter alia*). Such models initialized with contextualized word embeddings (Peters et al., 2018; Devlin et al., 2019) have achieved new state-of-the-art results for these tasks (Kitaev et al., 2019; Joshi et al., 2019b).

Since spans can have arbitrary length (i.e., number of tokens), fixed-dimensional span representations involve some form of (parameterized) pooling of the token representations. Existing models typically pick a *span representation* method (dashed boxes in Figure 1) that works well for the task(s) of interest. However, a comprehensive evaluation comparing various span representation methods across tasks is still lacking.



(a) Probing model for two-span tasks. This model can be used to decide whether two spans (here $[1, 2]$ and $[4, 4]$) are coreferent or not.

(b) Probing model for single-span tasks. This model can be used to decide whether a span (here $[1, 3]$) refers to a constituent or not.

Figure 1: Probing architectures for span representation methods. The models are very similar to that of Tenney et al. (2019b) but we explicitly separate the span representation part into a projection step followed by a choice among span representation methods.

In this work, we systematically compare and analyze a wide range of span representations (Section 3.2) by probing the representations via various NLP tasks, including constituent detection, constituent labeling, named entity labeling, semantic role labeling, mention detection, and coreference arc prediction (Section 2).[1] All of the tasks we consider naturally involve span representations. Similar comparisons are done by Tenney et al. (2019b), where they use this probing approach to compare several pretrained contextual embedding models, while keeping the span representation method fixed to self-attentive pooling (Lin et al., 2017; Lee et al., 2017). Here we vary both the choice of contextualized embedding models (among BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), XL-

[1]Code available at `https://github.com/shtoshni92/span-rep`

Net (Yang et al., 2019), and SpanBERT (Joshi et al., 2019a)) and the span representation methods. By analyzing the performance of each span representation method for multiple tasks, we aim to uncover the importance of choice of span representation.

We follow the "edge probing" setup of Tenney et al. (2019b) and introduce two new tasks to this setup, namely constituent detection and mention detection, which complement the constituent labeling and coreference arc prediction tasks, respectively, that are part of the original setup. For the full-scale comparison, we follow the original setup and keep the pretrained token representation models fixed, learning only layer weights and additional task-specific parameters on top of the weighted pretrained representations. We also conduct a small-scale study to compare the effect of fine-tuning on different span representations in terms of their relative ordering and for comparison with their non fine-tuned counterparts.

Overall, we find that the behavior of span representations tends to pattern according to whether they are based on information at the span boundaries versus using the entire span content. When pretrained models are frozen, we find that the choice of span representation is more important than the choice of pretrained model. When fine-tuning, the choice of span representation still has an impact on performance though it is much less pronounced than in the frozen case. Although the best-performing method can vary greatly among tasks, we find in general that a span representation that simply takes the max over time is a reliable choice across tasks.

2 Span Probing Tasks

We borrow four probing tasks applied by Tenney et al. (2019b), namely constituent labeling, named entity labeling, semantic role labeling, and coreference arc prediction. We also introduce two new tasks: constituent detection and mention detection. The specific tasks are described below.

Constituent labeling is the task of predicting the non-terminal label (e.g., noun phrase, verb phrase, etc.) for a span corresponding to a constituent.

Constituent detection is the task of determining whether a span of words corresponds to a constituent (i.e., a nonterminal node) in the constituency parse tree of the input sentence. We introduce this task as a complement to the task of constituent labeling, to further evaluate the syntactic ability of the span representation methods.

Named entity labeling (NEL) is the task of predicting the entity type of a given span corresponding to an entity, e.g., whether the span "German" in its sentence context refers to people, an organization, or a language.

Semantic role labeling (SRL) is concerned with predicting the semantic roles of phrases in a sentence. In this probing task the locations of the predicate and its argument are given, and the goal is to classify the argument into its specific semantic roles (`ARG0`, `ARG1`, etc.).

Mention detection is the task of predicting whether a span represents a mention of an entity or not. For example, in the sentence *"Mary goes to the market"*, the spans *"Mary"* and *"the market"* refer to mentions while all other spans are not mentions. The task is similar to named entity recognition (Tjong Kim Sang and De Meulder, 2003), but the mentions are not limited to named entities. We introduce this task as it is the first step for coreference resolution (Pradhan et al., 2012), if the candidate mentions are not explicitly given.

Coreference arc prediction is the task of predicting whether a pair of spans refer to the same entity. For example, in the sentence "John is his own enemy", "John" and "his" refer to the same entity.

3 Models

In this section, we first briefly describe the probing model, which is borrowed from Tenney et al. (2019b) with the extension to different span representations (Figure 1), followed by details of the various span representation methods we compare in this work.

3.1 Probing Model

The input to the model is a sentence $\mathbf{d} = \{d_1, \cdots, d_T\}$ where the d_i are tokens (produced by a tokenizer specific to a given choice of encoder). The sentence is first passed through a fixed, pretrained encoder, such as BERT, followed by a learned projection layer to obtain contextualized token embeddings $\{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_T\}$. These embeddings are then fed to span representation modules to get fixed-dimensional contextual span embeddings. Finally, the span embeddings are fed into a two-layer MLP followed by a sigmoid layer to predict the labels. For multiclass probing tasks with $|\mathcal{L}|$ labels, the predictions are made independently with separate MLPs per label resulting in a $[0,1]^{|\mathcal{L}|}$ vector. Finally, some tasks involve a single span, whereas others (coreference, semantic role labeling) involve two spans; in the latter case, the MLP takes as input the concatenation of the representations corresponding to the two spans.

3.2 Span Representation Methods

Given a span $s = [i, j]$ and its corresponding contextualized embeddings $[\boldsymbol{e}_i, \cdots, \boldsymbol{e}_j]$, where $\boldsymbol{e}_k \in \mathbb{R}^d$, a span representation module outputs a fixed-dimensional span representation $\boldsymbol{s}_{ij}$. Below we describe the various span representation methods compared in this work.

Average pooling is a simple average of the contextualized embeddings in the span window:

$$\boldsymbol{s}_{ij} = \frac{1}{j-i+1}\sum_{k=i}^{j}\boldsymbol{e}_k$$

Attention pooling or self-attention pooling is a learned weighted average over the contextualized token embeddings in the span:

$$\alpha_k = \boldsymbol{v}\cdot\boldsymbol{e}_k; \quad a_k = \frac{\exp(\alpha_k)}{\sum_{\ell=i}^{j}\exp(\alpha_\ell)}$$

$$\boldsymbol{s}_{ij} = \sum_{k=i}^{j} a_k \cdot \boldsymbol{e}_k$$

where $\boldsymbol{v}$ is a learned parameter vector. This pooling method is a popular choice for many NLP tasks (Lee et al., 2017; Lin et al., 2017), and is the one used by Tenney et al. (2019b).

Max pooling takes the maximum value over time for each dimension of the contextualized embeddings within the span. Max pooling has been frequently used to obtain fixed-dimensional sentence representations for classification tasks (Collobert et al., 2011; Hashimoto et al., 2017; Conneau et al., 2017).

Endpoint is a simple concatenation of the endpoints of the span: $\boldsymbol{s}_{ij} = [\boldsymbol{e}_i; \boldsymbol{e}_j]$. This is a popular choice for representing answer spans (Lee et al., 2016) in extractive question-answering tasks such as SQuAD (Rajpurkar et al., 2016). Note that in this case $\boldsymbol{s}_{ij} \in \mathbb{R}^{2d}$.

Diff-Sum is a variant of endpoint where we concatenate the sum and difference of the span endpoints: $\boldsymbol{s}_{ij} = [\boldsymbol{e}_j + \boldsymbol{e}_i; \boldsymbol{e}_j - \boldsymbol{e}_i]$. Diff-sum and its close variants have been used in parsing and SRL (Stern et al., 2017; Ouchi et al., 2018). As in endpoint, $\boldsymbol{s}_{ij} \in \mathbb{R}^{2d}$.

Coherent is a span representation proposed by Seo et al. (2019) for indexing phrases in a query-agnostic manner for question answering. First, the endpoints of the span are split into four parts:

$$\boldsymbol{e}_i = [\boldsymbol{e}_i^1; \boldsymbol{e}_i^2; \boldsymbol{e}_i^3; \boldsymbol{e}_i^4]$$

where $\boldsymbol{e}_i^1, \boldsymbol{e}_i^2 \in \mathbb{R}^a$ and $\boldsymbol{e}_i^3, \boldsymbol{e}_i^4 \in \mathbb{R}^b$, and therefore $2a + 2b = d$. The endpoints are then combined as:

$$\boldsymbol{s}_{ij} = [\boldsymbol{e}_i^1; \boldsymbol{e}_j^2; \boldsymbol{e}_i^3 \cdot \boldsymbol{e}_j^4]$$

where the last term $\boldsymbol{e}_i^3 \cdot \boldsymbol{e}_j^4$ is referred to as the *coherence* term; hence the name "coherent" (assigned by us). In this case, $\boldsymbol{s}_{ij} \in \mathbb{R}^{2a+1}$ where $2a + 1 < d$ since $2a + 2b = d$.[2]

4 Experimental Setup

4.1 Implementation details

All input strings are passed through contextual encoder models to obtain an embedding for each token. With frozen encoders the weighted average of outputs from all layers is used as the token representation $\boldsymbol{e}_k$ (Tenney et al., 2019b), while for fine-tuned encoders the last layer output is used unless otherwise stated. We investigate four pretrained models: BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), SpanBERT (Joshi et al., 2019a), and XLNet (Yang et al., 2019). Each has both "base" and "large" variants, and we experiment with both. Since some of the models, such as XLNet, only have cased versions, we use the cased version for all models. We use the HuggingFace (Wolf et al., 2019) implementation of the four models, which is based on PyTorch (Paszke et al., 2019).

Embeddings are first projected down to 256 dimensions.[3] For each span, a representation method (one of the methods from Section 3.2) is then applied to its sequence of projected vectors to obtain a fixed-length representation for the span. The span representations are concatenated (if there are more than one) and fed into a two-layer MLP followed by a sigmoid output layer. The two-layer MLP is a

[2]Seo et al. (2019) used a=480 and b=32 for 1024-dimensional BERT-large embeddings. We use the same proportions for the projected contextualized embeddings.

[3]For SRL we use separate projection matrices for the two spans involved in the task, as the two spans may require different types of information to be extracted. For all of the other tasks, a single projection matrix is used.

Task	Task Type	$\lvert\mathcal{L}\rvert$	#Examples (Train / Val. / Test)
Constituent labeling	Syntactic	30	1.9M / 255K / 191K
Constituent detection	Syntactic	2	3.1M / 426K / 318K
NEL	Semantic	18	128K / 20K / 13K
SRL	Semantic/Syntactic	66	599K / 83K / 62K
Mention detection	Syntactic	2	387K / 49K / 48K
Coreference arc prediction	Semantic	2	208K / 27K / 28K

Table 1: Dataset statistics for the six tasks.

stack of a linear layer, a non-linear layer with tanh activations, layer normalization, dropout (0.3 zeroing probability), and a second linear layer. The hidden dimension of the MLP is 256. Models are trained by minimizing binary cross-entropy loss against the set of true labels. Though some tasks (e.g., SRL) are multi-class classification, we make predictions for each label independently i.e. binary classification, which facilitates analysis on individual labels or label groups. The binary classification setting also allows using the micro-averaged F-score as the evaluation metric across tasks.

In experiments with frozen encoders, we only learn the encoder layer mixing weights, projection parameters, and MLP parameters, keeping the encoder parameters themselves fixed. For optimization we use Adam (Kingma and Ba, 2015) with an initial learning rate of 5×10^{-4} and a batch size of 64.[4] The model is evaluated on the validation set every 1000 steps and the learning rate is reduced by a factor of 2 if no improvement is seen in the previous 5 validation evaluations. Training stops if no improvement is seen for 20 validation evaluations.

In experiments with fine-tuning the encoders, we focus on only a subset of the frozen-encoder configurations for computational reasons. In particular, we only experiment with the "base" versions of BERT, RoBERTa, and SpanBERT.[5] All models are trained using Adam with an initial learning rate of 3×10^{-5} and a batch size of 64. Finally, the token embedding is either a layer-weighted combination or just the last layer.[6]

[4]We found non-trivial gains with this choice of higher learning rate compared to 1×10^{-4} used by Tenney et al. (2019b).

[5]We omit XLNet due to its relatively poor performance across tasks in the frozen setting.

[6]Typically the last layer embeddings perform slightly better but a few of those training runs failed and we present the layer-weighted results for those.

4.2 Data

Table 1 shows the dataset statistics of the six tasks evaluated in this study. For SRL, NEL, coreference arc prediction and constituent labeling, we use the annotations in the OntoNotes 5.0 corpus (Weischedel et al., 2013) and cast the original annotations into the edge probing format, following the same procedure as Tenney et al. (2019b) for pre-processing.

For the newly proposed constituent detection and mention detection tasks, we create our own datasets using the existing annotations and random negative sampling. For constituent detection, we use the constituent labeling annotations to get actual constituents, and for each constituent we sample a random negative span of the same length. We ensure that all negative spans are different and that we don't sample an actual constituent. We follow a similar procedure to get mention detection annotations from coreference arc prediction annotations. To make the mention detection task harder and more realistic, we sample 5 times more negatives than actual mentions.

5 Results

5.1 Results without Fine-Tuning

The results across tasks and models are shown in Figure 2. Overall we find max pooling to be the most robust and effective choice across tasks. Boundary-based span representations (i.e., ENDPOINT, DIFFSUM, COHERENT) are superior to entire-span methods (i.e., ATTN, MAX, AVG) on tasks which are more shallow/syntactic (e.g., constituent labeling and constituent detection), though max pooling is competitive with the boundary-based methods.

On the other hand, entire-span representations are good at semantic tasks like coreference arc prediction. As SRL has both semantic and syntactic characteristics, COHERENT, MAX, and ATTN show

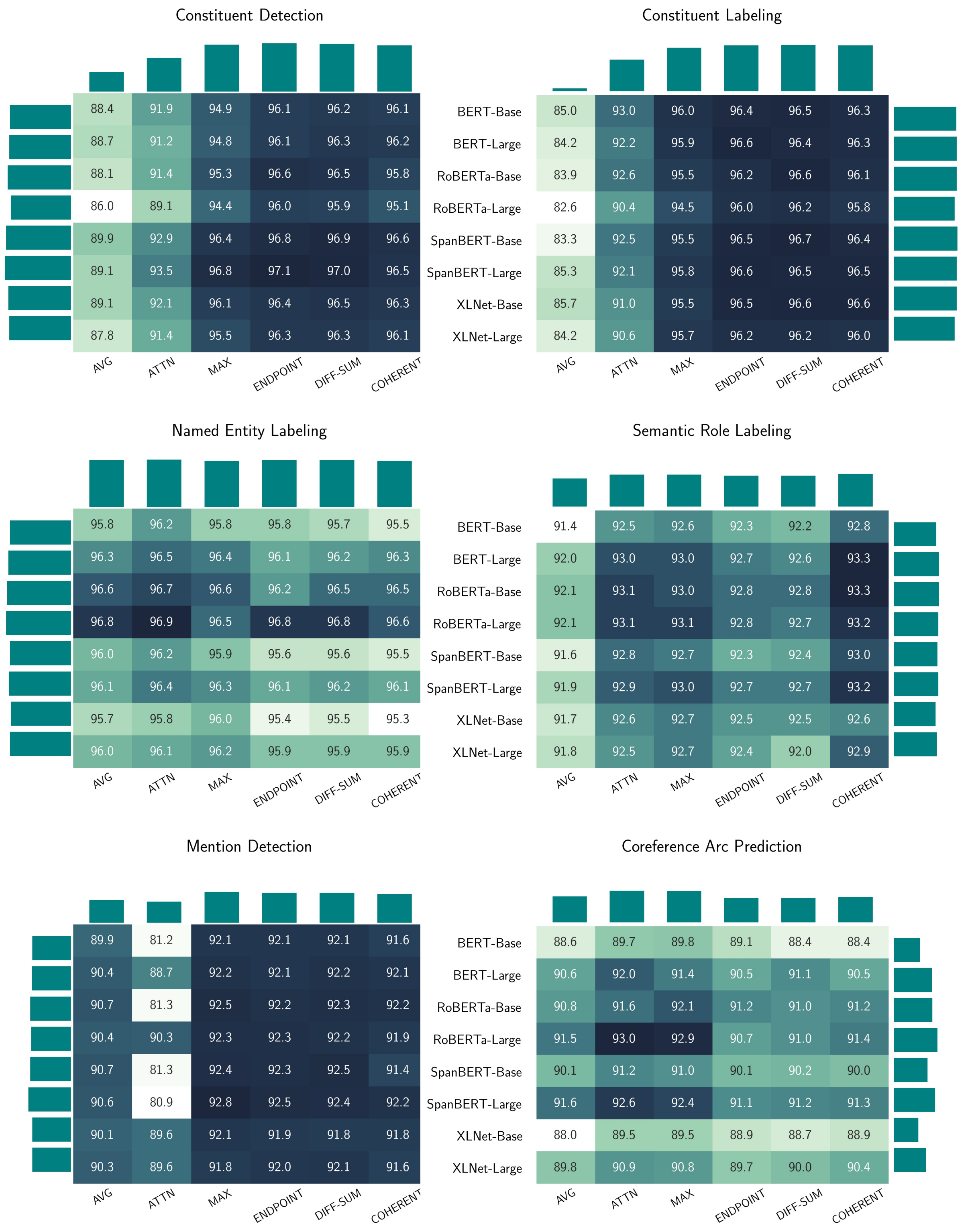


Figure 2: Results with frozen encoders (no fine-tuning) for the 6 different tasks presented as separate heatmap figures. Each heatmap represents the 48 combinations resulting from 6 span representations and 8 pretrained models. The bars at the side of the heatmap represent the max value in the row/column which is right below the bar.

similar performance with the other methods fairly close behind. We do not find large differences between span representation methods for NEL, which mainly contains short spans.

Model-wise, large models are usually better than base models though there exist exceptions (e.g., constituent labeling). RoBERTa shows strong performance across tasks. We also find that SpanBERT

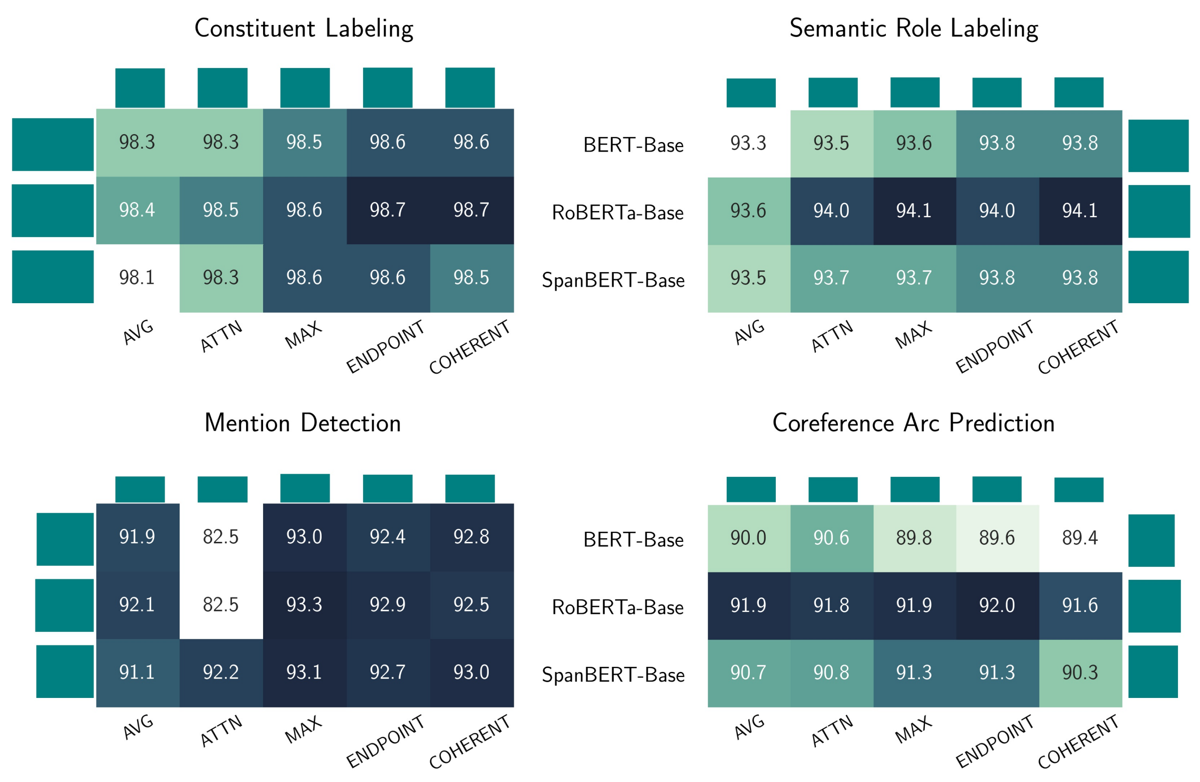


Figure 3: Results with fine-tuned encoders for four tasks, namely constituent labeling, SRL, mention detection, and coreference arc prediction, presented as separate heatmaps.

excels for tasks where boundary-based methods are superior, which may be because it is explicitly trained with an objective of predicting tokens inside a span given the boundary tokens.

Results for each task are summarized below.

Constituent detection/labeling: Boundary-based representations are better than entire-span ones, though MAX is close behind. Surprisingly, in these two tasks, large models are not as good as their base counterparts (Goldberg (2019) found similar exceptions for syntactic tasks).

Semantic role labeling: COHERENT is the best method on this task with MAX and ATTN being very close behind.

Mention detection and coreference arc prediction: ATTN and MAX perform the best for coreference arc prediction since they benefit from access to the entire span and thus to the semantic head of the span (Lee et al., 2017). For mention detection the trends are reversed, except for MAX, with the boundary-based methods doing quite well. This is not surprising since the mention detection task is somewhat close to constituent tasks. Surprisingly, ATTN shows high variance across models and performs worse than even AVG. Since we initialize the attention parameter vector $\boldsymbol{v}$ to all zeroes, this result means that not learning the attention vector is surprisingly better than learning them.[7] Preliminary investigation of the learned attention weights did not provide any clues.

Mention detection and coreference arc prediction together complete the pipeline for coreference resolution. The preference for different forms of span representations between the two (except for MAX) suggests that different span representations can be considered for different stages of the coreference resolution task. Interestingly, one of the best performing end-to-end coreference models (Lee et al., 2017) uses a concatenation of a boundary-based span representation, ENDPOINT, and ATTN.

Some of our observations may be confounded with training set sizes, which vary from coreference arc prediction on the small end (208K) up to constituent tasks on the largest end (1.9M), with SRL (599K) in the middle of the range.

5.2 Results with Fine-Tuning

State-of-the-art models almost always fine-tune the pretrained encoders. However, the training is quite

[7]Stopping the gradient for the attention vector indeed performed similarly to AVG.

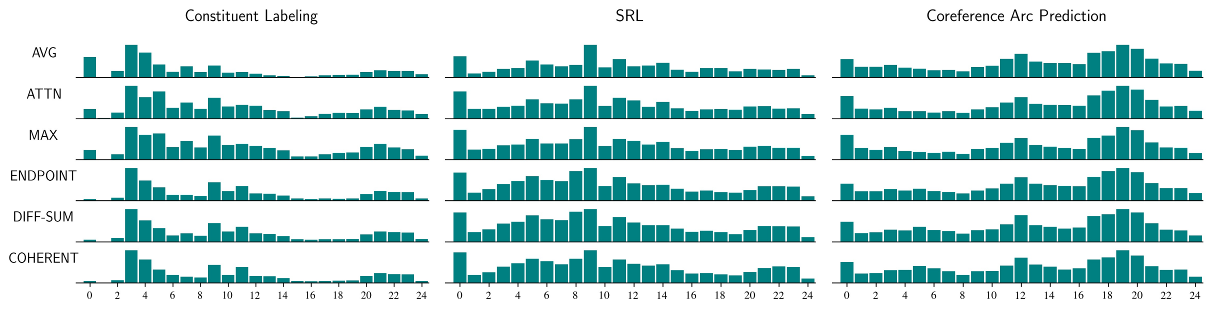


Figure 4: Visualization of layerwise weights learned for all span representation methods for constituent labeling, SRL, and coreference arc prediction for the RoBERTa-large model.

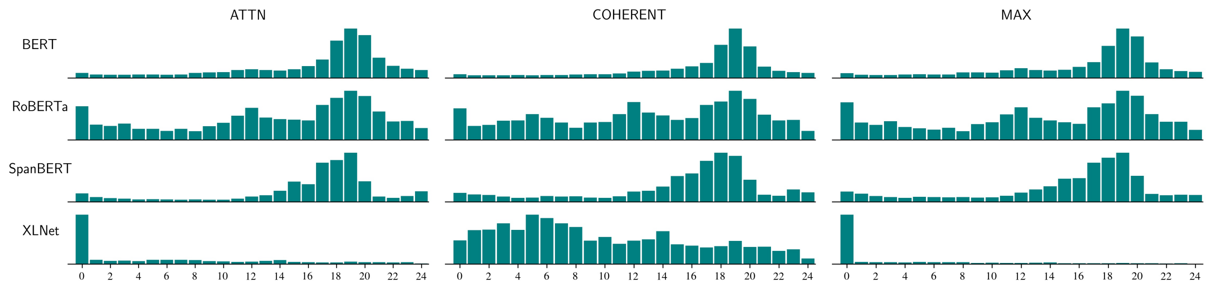


Figure 5: Visualization of layerwise weights learned for the coreference arc prediction task for three span representation methods with {BERT, RoBERTa, SpanBERT, XLNet}-large models. While BERT, SpanBERT, and XLNet have peaked weight distributions, RoBERTa's weights are more spread out. Oddly enough XLNet chooses to place the most weight on the embedding layer for ATTN and MAX (the two best span representations for XLNet-large).

computationally expensive; hence, we perform the span representation comparison for a small set of the total configurations whose results are shown in Figure 3. In general, fine-tuning improves the results for all span representation methods across tasks, with the performance of different span representation methods now more tightly clustered. This is best illustrated by the constituent labeling task where without fine-tuning AVG trails by 10+ F-score with respect to the best span representation but less than 0.5 F-score with fine-tuning.

6 Analysis

In this section we analyze the impact of span representation method with fixed pretrained encoders.

6.1 Layerwise Weight Analysis

Figures 4 and 5 visualize learned layer weights for different task, span representation, and pretrained encoder combinations. Within a model and task, we generally found that the layer weights were fairly consistent across span representation methods. Overall, we find similar trends to prior work in analyzing layer weights for downstream NLP tasks, namely that constituency parsing has higher weight for lower layers and coreference has most weight on higher layers, with SRL in between (Liu et al., 2019a; Tenney et al., 2019a). For ablation analysis of layer weights, whether to learn them or not, see Appendix A.1.

6.2 Label-Specific Analysis of Span Groups

We seek to determine whether boundary-based span representation methods (COHERENT, DIFFSUM, and ENDPOINT) differ systematically from methods that consider the entire span (ATTN, MAX, and AVG). We pooled predictions from the three methods in each group for RoBERTa-large and calculated the recall for particular labels, for two tasks: SRL and constituent labeling (analysis for NEL appears in Appendix A.2). We found the labels with the largest differences in recall between the two groups, and discuss our findings below.

SRL. Table 2 shows the argument labels with the largest differences in recall (ΔR) between the two groups, limiting our analysis to the 20 most frequent argument labels. Labels with positive ΔR are handled better by boundary-based methods. These tend to be arguments that can be identified based on particular words, often function words, at the boundaries. For example, ARGM-DIS is found

Argument	ΔR	Notes	Examples
C-ARG1 (continuous argument)	10.85	often starts with *to*, similar to `xcomp` in universal dependencies	"were granted the right earlier this year **to ship sugar**", "brought more money into a city **than it took out**", "Food prices are expected **to be unchanged**"
ARGM-DIS (discourse)	3.24	mostly single-word discourse modifiers like *and* or *but*	"**In addition** , the government is figuring", "**But** eluding", "**And** the USIA said", "**of course** , there 's that word"
ARGM-MNR (manner)	2.03	mostly adverbs and prepositional phrases	"...get married **in a tuxedo**", "**relatively** respected", "Moving **rapidly** through school", "...do n't leave home **without the American Express card**"
ARG3 (starting point, benefactive, attribute)	-4.28	multiple functions depending on predicate, and thus a variety of boundary words	"And what I had mentioned **about my mother bugging me** was...", "You should feel comfortable staying **there**", "...he believes that it can bring the market **back up** after a plunge", "Biologists mixed a mold element in the cells of plants with pearl powder **to produce a granulated drug**"
ARGM-DIR (direction)	-5.20	typically a word or short phrase, mostly adverbs, prepositional phrases, particles, and adjectives	"newspapers turning **to color** on their pages", "bond prices rapidly turned **south**", "major brokerage firms rushed **out** ads", "takes the dispute **to the Supreme Court**", "we have to get **out of bed**", "toss the chalk **back and forth**"
ARGM-EXT (extent)	-8.14	often a short phrase like *more*, *very much*, *a lot*, etc.; limited semantics, range of surface forms	"you 're critical to yourself **too much**", "of freezing , **at least partially**", "increase **of 32 %**", "life has changed **a lot**", "Thank you **very much**"

Table 2: Analysis of argument labels for semantic role labeling. ΔR = argument recall% with boundary-based span representation methods minus recall% with entire-span methods. In the examples, predicates are underlined and arguments of the given type are shown in boldface.

Label	ΔR	Examples
SBAR	6.1	"that are missing", "who owned the land"
PRN	4.6	", she says ,", ", it turns out ,", "(file photo)", "(hey , it 's possible)"
ADJP	3.3	"liable", "available to anyone", "more generous", "satisfied with where they work", "at least somewhat interesting"
PP	2.7	"in 1966", "within a community"
SBARQ	-6.1	"What can we do ?", "So what should be done .", "and what is money for", "how shall I say"
SQ	-6.5	"Did you see ?", "will I do now", "do you make of", "You still building"
FRAG	-6.5	"Or something .", "well below 1988 activity", "As for Mr. Papandreou ?"
SINV	-15.7	"should the Air Force order the craft", "say Mr. Dinkins 's managers", "notes Huang frankly", "invest they will"

Table 3: Analysis of labels for constituent labeling. ΔR = label recall% with ENDPOINT minus recall% with MAX. We restrict this analysis to labels that appear at least 100 times in our development set.

mostly with single-word modifiers in this dataset (like *and* and *but*). Arguments that are handled better by entire-span methods are more diverse in terms of their boundary words. ARGM-EXT is used for arguments with relatively limited semantics (as shown in the examples) but a variety of surface realizations.

Constituent labeling. Table 3 shows a similar analysis for constituent labeling, though in this case we compare only a single method from each family: ENDPOINT and MAX. We do this because MAX is comparable in performance to the boundary methods while ATTN and AVG are significantly worse. We choose ENDPOINT as our single representative of the boundary methods in order to compare only two methods, though we found the same trends for others in its group.

ENDPOINT has higher recall on several labels, shown in the top part of the table. There is a 6% difference for SBAR, which is a clause introduced by a (possibly empty) subordinating conjunction. About 25% of SBAR constituents begin with *that*, and many others start with some other very common subordinating conjunction, making SBAR easier to find for methods that focus on boundary words. Parentheticals (PRN) frequently begin and end with commas or parentheses. ADJPs typically begin or end with an adjective and PPs nearly always begin with prepositions.

The lower part of Table 3 shows labels where MAX has higher recall than ENDPOINT. The largest difference is in SINV, which is an "inverted" declarative sentence, that is, a sentence in which the subject follows the conjugated verb. These often look like VPs based on boundary words but are more diverse syntactically; a few short examples are shown in the table. The other labels also show syntactic diversity. FRAG (fragment) has many realizations that vary widely in terms of their syntax. while SBARQ and SQ often start with *wh*-words and end in question marks, they show significant variation.

7 Related Work

Many of the span representations that we consider here were proposed previously for specific tasks, such as the attention-weighted pooling of Lee et al. (2017) for coreference resolution; the endpoint-based representation of Lee et al. (2016) and the "coherent" endpoint-based representation of Seo et al. (2019) for question answering; and combinations of differences and sums of endpoint representations for parsing and semantic role labeling (Stern et al., 2017; Ouchi et al., 2018). These are described in more detail in Section 3.2.

Other recent work has considered pooling approaches such as the difference between endpoint representations (Wang and Chang, 2016; Cross and Huang, 2016) or a concatenation of endpoint and attention-based representations (Li et al., 2016). Other approaches concatenate additional specialized feature vectors, such as the span length or position information (Lee et al., 2017; He et al., 2018; Kuribayashi et al., 2019). Some work has also considered explicitly composing span representations via syntactic parse trees, such as recursive neural networks (Li et al., 2014), and some unsupervised parsing models produce span representations as a byproduct of training (Drozdov et al., 2019; Shi et al., 2019).

At the same time, there has been significant effort devoted to the related problem of learning representations for sentences or even longer texts (Kalchbrenner et al., 2014; Iyyer et al., 2015; Kiros et al., 2015; Wieting et al., 2016; Conneau et al., 2017; Shen et al., 2018, *inter alia*). Much of this work focuses on pooling over word representations, often finding that simple pooling operations like averaging perform surprisingly well (Wieting et al., 2016; Shen et al., 2018). Shen et al. (2018) did a similar empirical study to ours in spirit, comparing a variety of pooling models for sentence representations across tasks.

In this work we are mainly focusing on the models for computing span representations given pretrained token embeddings, but we also include a variety of pretrained contextual embeddings. One in particular, SpanBERT (Joshi et al., 2019a), was designed to enable improved span representations. While recent work has compared across pretrained contextual embeddings for representing spans (Tenney et al., 2019b), to our knowledge there has been no systematic comparison of methods for combining these contextual embeddings into span representations across a variety of tasks.

8 Conclusion

We systematically compared multiple span representation methods, combined with various base embedding models, on various tasks. Our analysis includes two new tasks that we propose to tease apart different aspects of span representations. When using fixed, pretrained encoders, we find that, although max pooling is a fairly reliable representation across tasks, the optimal span representation varies with respect to the syntactic and semantic nature of the task. Finally, fine-tuning reduces the impact of span representation choice on performance, though it involves significant computational expense. Our results are likely to be most useful for those without the computational capabilities to perform fine-tuning of large pretrained encoders, in which case there are significant differences among methods.

References

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *JMLR*, 12:2493–2537.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *EMNLP*.

James Cross and Liang Huang. 2016. Span-Based Constituency Parsing with a Structure-Label System and Provably Optimal Dynamic Oracles. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised Latent Tree Induction with Deep Inside-Outside Recursive Auto-Encoders. In *NAACL-HLT*.

Yoav Goldberg. 2019. Assessing BERT's Syntactic Abilities. *arXiv*, abs/1901.05287.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *EMNLP*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly Predicting Predicates and Arguments in Neural Semantic Role Labeling. In *ACL*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL-IJCNLP*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2019a. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *TACL*.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019b. BERT for Coreference Resolution: Baselines and Analysis. In *EMNLP-IJCNLP*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *ACL*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NeurIPS*.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual Constituency Parsing with Self-Attention and Pre-Training. In *ACL*.

Nikita Kitaev and Dan Klein. 2018. Constituency Parsing with a Self-Attentive Encoder. In *ACL*.

Tatsuki Kuribayashi, Hiroki Ouchi, Naoya Inoue, Paul Reisert, Toshinori Miyoshi, Jun Suzuki, and Kentaro Inui. 2019. An Empirical Study of Span Representations in Argumentation Structure Parsing. In *ACL*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end Neural Coreference Resolution. In *EMNLP*.

Kenton Lee, Tom Kwiatkowski, Ankur P. Parikh, and Dipanjan Das. 2016. Learning Recurrent Span Representations for Extractive Question Answering. *CoRR*, abs/1611.01436.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive Deep Models for Discourse Parsing. In *EMNLP*.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse Parsing with Attention-based Hierarchical Neural Networks. In *EMNLP*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic Knowledge and Transferability of Contextual Representations. In *NAACL-HLT*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa:A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A Span Selection Model for Semantic Role Labeling. In *EMNLP*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*.

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

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes. In *EMNLP and CoNLL - Shared Task*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-Time Open-Domain Question Answering with Dense-Sparse Phrase Index. In *ACL*.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. In *ACL*.

Haoyue Shi, Jiayuan Mao, Kevin Gimpel, and Karen Livescu. 2019. Visually grounded neural syntax acquisition. In *ACL*.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A Minimal Span-Based Neural Constituency Parser. In *ACL*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT Rediscovers the Classical NLP Pipeline. In *ACL*.

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

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *NAACL*.

Wenhui Wang and Baobao Chang. 2016. Graph-based Dependency Parsing with Bidirectional LSTM. In *ACL*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, and Michelle Franchini et al. 2013. OntoNotes release 5.0 LDC2013T19. *Linguistic Data Consortium*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards Universal Paraphrastic Sentence Embeddings. In *ICLR*.

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.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*.

A Appendix

A.1 Mixing Weights for Layers

Model	AVG	ATT	MAX	EP	DS	COH
BERT-large	90.6	92	91.4	90.5	91.1	90.5
-mix_wt	90.3	91.3	91.5	90.4	90.4	90.3
RoBERTa-large	91.5	93	92.9	90.7	91.0	91.4
-mix_wt	91.0	92.7	92.6	90.9	90.7	90.6
SpanBERT-large	91.6	92.6	92.4	91.1	91.2	91.3
-mix_wt	90.4	91.4	91.5	90.6	90.3	90.2
XLNet-large	89.8	90.9	90.8	89.7	90.0	90.4
-mix_wt	89.4	90.7	90.8	89.5	90.0	90.7

Table 4: Analysis of importance of learning mixing weights for combination of different models and span representations for the coreference arc prediction task.

In the table above we analyze the effect of learning the layerwise mixing weights vs simple averaging over layers in context of the coreference arc prediction task. ATTN-based models suffer the biggest drop with a drop of 0.6% absolute on average. Among pretrained contextual embedding models, SpanBERT-large is hurt the most with a drop of 1% absolute on average. Surprisingly, XLNet drops by only 0.1% on average even though its attention plots looked quite peaky for some of the span representations.

A.2 Label-Specific Analysis of Span Groups for NEL

Label	ΔR	Examples
ORDINAL	1.2	"first", "second", "First", "6th", "ninth"
CARDINAL	0.3	"two", "10", "Dozens", "at least 37"
TIME	-2.3	"seven o'clock", "two hours", "about ten", "eight fifty in the morning"
LAW	-2.9	"Paragraph 14 of Article 19", "the Geneva Convention", "Dru 's Law"
LOC	-3.2	"the Sierra Nevada Mountains", "Asia", "Mai Po Marshes"
WORK_OF_ART	-6.0	"The End of the Day", "Carry On Trading", "News Night Tonight"

Table 5: Analysis of labels for NEL. ΔR = label recall% with boundary-based span representation methods minus recall% with entire-span methods.

NEL. Table 5 shows a similar analysis for entity labeling as done in Section 6.2. The labels with higher recall under the boundary-based methods are limited to ORDINAL and CARDINAL numbers, which tend to be very short and highly regular (nearly all ORDINAL entities are one token and approximately half are *first*). The entire-span methods achieve much higher recall for the WORK_OF_ART label, and also for LOC, LAW, and TIME. These entities tend to be multi-word phrases with a variety of syntactic forms and without consistent boundary words.

It may be surprising that ORDINAL is better detected by the boundary methods, since nearly all ORDINAL entities are a single token, and the entire-span methods reduce to a simple form for single tokens. However, this may show that the entire-span methods are being trained to abstract over the contents of the span, thereby losing some of the surface information. The boundary-based methods, by contrast, devote particular parts of the span representation to the boundary position representations, thereby providing a more direct/explicit connection between those boundary words and the downstream classifier.