WHAT DO YOU LEARN FROM CONTEXT? PROBING FOR SENTENCE STRUCTURE IN CONTEXTUALIZED WORD REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Contextualized representation models such as CoVe (McCann et al., 2017) and ELMo (Peters et al., 2018a) have recently achieved state-of-the-art results on a diverse array of downstream NLP tasks. Building on recent token-level probing work, we introduce a novel *edge probing* task design and construct a broad suite of sub-sentence tasks derived from the traditional structured NLP pipeline. We probe word-level contextual representations from three recent models and investigate how they encode sentence structure across a range of syntactic, semantic, local, and long-range phenomena. We find that ELMo encodes linguistic structure at the word level better than other comparable models, and that existing models trained on language modeling and translation produce strong representations for syntactic phenomena, but only offer small improvements on semantic tasks over a non-contextual baseline.

1 INTRODUCTION

Pretrained word embeddings (Mikolov et al., 2013; Pennington et al., 2014) are a staple tool for NLP. These models provide continuous representations for word types, typically learned from co-occurrence statistics on unlabeled data, and improve generalization of downstream models across many domains. Recently, a number of models have been proposed for *contextualized* word embeddings. Instead of using a single, fixed vector per word type, these models run a pretrained encoder network over the sentence to produce contextual embeddings of each token. The encoder, usually an LSTM (Hochreiter & Schmidhuber, 1997) or a Transformer (Vaswani et al., 2017), can be trained on objectives like machine translation (McCann et al., 2017) or language modeling (Peters et al., 2018a; Radford et al., 2018; Howard & Ruder, 2018), for which large amounts of data are available. The activations of this network–a collection of one vector per token–fit the same interface as conventional word embeddings, and can be used as a drop-in replacement input to any model. Applied to popular models, this technique has yielded significant improvements to the state-of-the-art on several tasks, including constituency parsing (Kitaev & Klein, 2018), semantic role labeling (He et al., 2018; Strubell et al., 2018), and coreference (Lee et al., 2018), and has outperformed competing techniques (Kiros et al., 2015; Conneau et al., 2017) that produce fixed-length representations for entire sentences.

Our goal in this work is to understand where these contextual representations improve over conventional word embeddings. Recent work has explored many token-level properties of these representations, such as their ability to capture part-of-speech tags (Blevins et al., 2018; Belinkov et al., 2017b; Shi et al., 2016), morphology (Belinkov et al., 2017a;b), or word-sense disambiguation (Peters et al., 2018a). Peters et al. (2018b) extends this to constituent phrases, and present a heuristic for unsupervised pronominal coreference. We expand on this even further and introduce a suite of *edge probing* tasks covering a broad range of syntactic, semantic, local, and long-range phenomena. In particular, we focus on asking what information is encoded at each position, and how well it encodes structural information about that word's role in the sentence. Is this information primarily syntactic in nature, or do the representations also encode higher-level semantic relationships? Is this information local, or do the encoders also capture long-range structure?

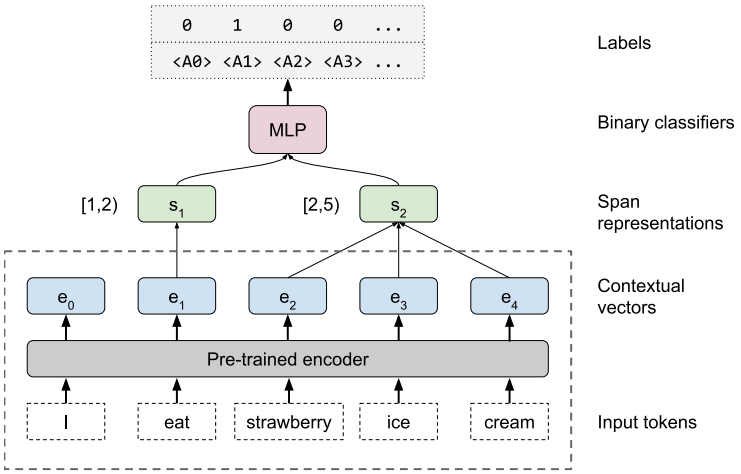

Figure 1: Probing model architecture (§ 3.1). All parameters inside the dashed line are fixed, while we train the span pooling and MLP classifiers to extract information from the contextual vectors. The example shown is for semantic role labeling, where $s^{(1)} = [1, 2)$ corresponds to the predicate ("eat"), while $s^{(2)} = [2, 5)$ is the argument ("strawberry ice cream"), and we predict label `A1` as positive and others as negative. For entity and constituent labeling, only a single span is used.

We approach these questions with a probing model (Figure 1) that sees only the contextual embeddings from a fixed, pretrained encoder. The model can access only embeddings within given spans, such as a predicate-argument pair, and must predict properties, such as semantic roles, which typically require whole-sentence context. We use data derived from traditional structured NLP tasks: tagging, parsing, semantic roles, and coreference. Common corpora such as OntoNotes (Weischedel et al., 2013) provide a wealth of annotations for well-studied concepts which are both linguistically motivated and known to be useful intermediates for high-level language understanding. We refer to our technique as "edge probing", as we decompose each structured task into a set of graph edges (§ 2) which we can predict independently using a common classifier architecture (§ 3.1).[1] We probe three popular contextual representation models (§ 3.2): CoVe (McCann et al., 2017), ELMo (Peters et al., 2018a), and the recently-released Transformer LM (Radford et al., 2018). We focus on these models because their pretrained weights and code are available, since these are most likely to be used by researchers. We compare to word-level baselines to separate the contribution of context from lexical priors, and experiment with augmented baselines to better understand the role of pretraining and the ability of encoders to capture long-range dependencies.

2 Edge Probing

To carry out our experiments, we define a novel "edge probing" framework motivated by the need for a uniform set of metrics and architectures across tasks. Our framework is generic, and can be applied to any task that can be represented as a labeled graph anchored to spans in a sentence.

Formulation. Formally, we represent a sentence as a list of tokens $T = [t_0, t_1, \ldots, t_n]$, and a labeled edge as $\{s^{(1)}, s^{(2)}, L\}$. We treat $s^{(1)} = [i^{(1)}, j^{(1)})$ and, optionally, $s^{(2)} = [i^{(2)}, j^{(2)})$ as (end-exclusive) spans. For unary edges such as constituent labels, $s^{(2)}$ is omitted. We take L to be a set of zero or more targets from a task-specific label set $\mathcal{L}$.

To cast all tasks into a common classification model, we focus on the *labeling* versions of each task. Spans (gold mentions, constituents, predicates, etc.) are given as inputs, and the model is trained to predict L as a multi-label target. We note that this is only one component of the common pipelined (or end-to-end) approach to these tasks, and that in general our metrics are not comparable to models that jointly perform span *identification* and labeling. However, since our focus is on analysis rather than application, the labeling version is a better fit for our goals of isolating individual phenomena of interest, and giving a uniform metric – binary F1 score – across our probing suite.

[1] Our code is publicly available at `https://to.be.released/after/review`.

POS	The important thing about Disney is that it is a global [brand]$_1$. $\rightarrow$ NN (Noun)
Constit.	The important thing about Disney is that it [is a global brand]$_1$. $\rightarrow$ VP (Verb Phrase)
Depend.	[Atmosphere]$_1$ is always [fun]$_2$ $\rightarrow$ nsubj (nominal subject)
NE	The important thing about [Disney]$_1$ is that it is a global brand. $\rightarrow$ Organization
SRL	[The important thing about Disney]$_2$ [is]$_1$ that it is a global brand. $\rightarrow$ Arg1 (Agent)
SPR	[It]$_1$ [endorsed]$_2$ the White House strategy... $\rightarrow$ {awareness, existed_after, ...}
Coref. (O)	The important thing about [Disney]$_1$ is that [it]$_2$ is a global brand. $\rightarrow$ True
Coref. (W)	[Characters]$_2$ entertain audiences because [they]$_1$ want people to be happy. $\rightarrow$ True Characters entertain [audiences]$_2$ because [they]$_1$ want people to be happy. $\rightarrow$ False

Table 1: Example sentence, spans, and target label for each task. O = OntoNotes, W = Winograd.

Tasks. Our experiments focus on seven core NLP labeling tasks: part-of-speech, constituents, dependencies, named entities, semantic roles, coreference, and semantic proto-roles. The tasks and their respective datasets are described below, and also detailed in Table 1 and Appendix A.

Part-of-speech tagging (POS) is the syntactic task of assigning tags such as noun, verb, adjective, etc. to individual tokens. We let $s_1 = [i, i+1)$ be a single token, and seek to predict the POS tag.

Constituent labeling is the more general task concerned with non-terminal label for a span of tokens within the phrase-structure parse of the sentence: e.g. is the span a noun phrase, a verb phrase, etc. We let $s_1 = [i, j)$ be a known constituent, and seek to predict the constituent label.

Dependency labeling is similar to constituent labeling, except that rather than aiming to position a span of tokens within the phrase structure, dependency labeling seeks to predict the functional relationships of one token relative to another: e.g. is in a modifier-head relationship, a subject-object relationship, etc. We take $s_1 = [i, i+1)$ to be a single token and $s_2 = [j, j+1)$ to be its syntactic head, and seek to predict the dependency relation between tokens i and j.

Named entity (NE) labeling is the task of predicting the category of an entity referred to by a given span, e.g. does the entity refer to a person, a location, an organization, etc. We let $s_1 = [i, j)$ represent an entity span and seek to predict the entity type.

Semantic role labeling (SRL) is the task of imposing predicate-argument structure onto a natural language sentence: e.g. given a sentence like *"Mary pushed John"*, SRL is concerned with identifying *"Mary"* as the pusher and *"John"* as the pushee. We let $s_1 = [i_1, j_1)$ represent a known predicate and $s_2 = [i_2, j_2)$ represent a known argument of that predicate, and seek to predict the role that the argument s_2 fills–e.g. pusher (agent/`ARG1`) vs. pushee (patient/`ARG2`).

Coreference–specifically, entity coreference–is the task of determining whether two spans of tokens ("mentions") refer to the same entity: e.g. in a given context, do *"Obama"* and *"the former president"* refer to the same person, or do *"New York City"* and *"there"* refer to the same place. We let s_1 and s_2 represent known mentions, and seek to make a binary prediction of whether they co-refer.

Semantic proto-role (SPR) labeling requires adding finer-grained, non-exclusive annotations on top of core semantic role relations, such as *change of state* or *awareness*. E.g. given then sentence *"Mary pushed John"*, whereas SRL is concerned with identifying *"Mary"* as the pusher, SPR is concerned with identifying attributes such as *awareness* (whether the pusher is *aware* that they are doing the pushing). We let s_1 represent a predicate span and s_2 a known argument head, and perform a multi-label classification over potential attributes of the predicate-argument relation.

Datasets. We use the annotations in the OntoNotes 5.0 corpus (Weischedel et al., 2013) for five of the above seven tasks: POS tags, constituents, named entities, semantic roles, and coreference. In all cases, we simply cast the original annotation into our edge probing format. For POS tagging, simply extract these labels from the constituency parse data in OntoNotes. For coreference, since OntoNotes only provides annotations for positive examples (pairs of mentions that corefer) we generate negative examples by generating all pairs of mentions that are not explicitly marked as coreferent.

The OntoNotes corpus does not contain annotations for dependencies or for proto-roles. Thus, for dependencies, we use the English Web Treebank portion of the Universal Dependencies 1.2 release (Nivre et al., 2015). For SPR, we use two datasets, one (SPR1; Teichert et al. (2017)) derived from Penn Treebank and one (SPR2; (Rudinger et al., 2018)) derived from English Web Treebank.

In addition to the OntoNotes coreference examples, we include an extra "challenge" coreference dataset based on the Winograd schema (Levesque et al., 2012). Winograd schema problems focus on cases of pronoun resolution which are syntactically ambiguous and thus are intended to require subtler semantic inference in order to resolve correctly (see example in Table 1). We use the version of the Definite Pronoun Resolution (DPR) dataset (Rahman & Ng, 2012) employed by White et al. (2017), which contains balanced positive and negative pairs.

3 Experimental Set-Up

3.1 Probing Model

Our probing architecture is illustrated in Figure 1. The model is designed to have limited expressive power on its own, as to focus on what information can be extracted from the contextual embeddings. We take a list of contextual vectors $[e_0, e_1, \ldots, e_n]$ and integer spans $s^{(1)} = [i^{(1)}, j^{(1)})$ and (optionally) $s^{(2)} = [i^{(2)}, j^{(2)})$ as inputs, and use a projection layer followed by the self-attention pooling operator of Lee et al. (2017) to compute fixed-length span representations. Pooling is only within the bounds of a span, e.g. the vectors $[e_i, e_{i+1}, \ldots, e_{j-1}]$, which means that the only information our model can access about the rest of the sentence is that provided by the contextual embeddings.

The span representations are concatenated and fed into a two-layer MLP followed by a sigmoid output layer. We train by minimizing binary cross-entropy against the target label set $L \in \{0, 1\}^{|\mathcal{L}|}$. Our code is implemented in PyTorch (Paszke et al., 2017) using the AllenNLP (Gardner et al., 2018) toolkit. For further details on training, see Appendix B.

3.2 Sentence Representation Models

We explore three recent contextual encoder models: CoVe, ELMo, and the Transformer LM. Each model takes tokens $[t_0, t_1, \ldots, t_n]$ as input and produces a list of contextual vectors $[e_0, e_1, \ldots, e_n]$.

CoVe (McCann et al., 2017) uses the top-level activations of a two-layer biLSTM trained on English-German translation, concatenated with 300-dimensional GloVe vectors. The source data consists of 7 million sentences from web crawl, news, and government proceedings (WMT 2017; Bojar et al. (2017)).

ELMo (Peters et al., 2018a) is a two-layer bidirectional LSTM language model, built over a context-independent character CNN layer and trained on the Billion Word Benchmark dataset (Chelba et al., 2014), consisting primarily of newswire text. We follow standard usage and take a linear combination of the ELMo layers, using learned task-specific scalars (Equation 1 of Peters et al., 2018a).

The **Transformer LM** of Radford et al. (2018) is a 12-layer Transformer (Vaswani et al., 2017) encoder trained as a left-to-right language model on the Toronto Books Corpus (Zhu et al., 2015). Departing from the original authors, we do not fine-tune the encoder[2], and we concatenate the learned subword embeddings with the top-layer activations of the model.

The resulting contextual vectors have dimension $d = 900$ for CoVe, $d = 1024$ for ELMo, and $d = 768 + 768 = 1536$ for the Transformer LM.[3] The pretrained models expect different tokenizations and input processing. We use a heuristic alignment algorithm, explained in detail in Appendix D, based on byte-level Levenshtein distance in order to re-map spans from the source data to the tokenization expected by the above models.

[2]We note that there may be information not easily accessible without fine-tuning the LSTM weights. This can be easily explored within our framework, e.g. using the techniques of Howard & Ruder (2018) or Radford et al. (2018). We leave this to future work, and hope that our code release will facilitate such continuations.

[3]For further details, see Appendix C.

4 EXPERIMENTS

Again, we want to answer: What do contextual representations encode that conventional word embeddings do not? Our experimental comparisons, described below, are intended to ablate various aspects of contextualized encoders in order to illuminate how the model captures different types of linguistic information.

Lexical Baselines. In order to probe the effect of each *contextual* encoder , we train our probing models directly on the most closely related context-independent word representations. This baseline measures the performance that can be achieved from lexical priors alone, without any access to surrounding words. For CoVe, we compare to the embedding layer of that model, which are 300-dimensional GloVe vectors trained on 840 billion tokens of CommonCrawl (web) text. For ELMo, we use the activations of the context-independent character-CNN layer (layer 0) from the full model. For the Transformer LM, we use the learned subword embeddings from the full model.

Randomized ELMo. Randomized neural networks have recently (Zhang & Bowman, 2018) shown surprisingly strong performance on many tasks, suggesting that architecture may play a significant role in learning useful feature functions. To help understand what is actually *learned* during the encoder pretraining, we compare with a version of the ELMo model in which all weights above the lexical layer (layer 0) are replaced with random orthonormal matrices[4].

Word-Level CNN. To what extent do contextual encoders capture long-range dependencies, versus simply modeling local context? We extend our lexical baseline by introducing a fixed-width convolutional layer on top of the word representations. As comparing to the lexical baseline factors out word-level priors, comparing to this CNN baseline factors out local relationships, such as the presence of nearby function words, and allows us to see the contribution of long-range context to encoder performance. To implement this, we replace the projection layer in our probing model with a fully-connected CNN that sees ± 1 or ± 2 tokens around the center word (i.e. kernel width 3 or 5).

5 RESULTS

Using the above experimental design, we return to the central questions originally posed. That is, what types of syntactic and semantic information does each model encode at each position? And is the information captured primarily local, or do contextualized embeddings encode information long-range sentential structure?

Comparison of representation models. We report F1 scores for ELMo, CoVe, and the Transformer LM in Table 2. We observe that ELMo outperforms the Transformer LM by a small margin (1.1 F1 points on average) on most tasks[5] and CoVe by a somewhat larger margin (5.2 F1 points on average), meaning that the information in ELMo's word representations makes it easier to recover details of sentence structure than does the information presented by the other models. It is important to note that while ELMo, CoVe, and the Transformer LM can be applied to the same problems, they differ in architecture, training objective, and both the quantity and genre of training data (§ 3.2). Furthermore, on all tasks except for Winograd coreference, the lexical representations used by the ELMo and Transformer LM models outperform GloVe vectors (by 5.4 and 2.4 points, respectively). This is particularly pronounced on constituent and semantic role labeling, where the model may be benefiting from better handling of morphology by character-level or subword representations.

Genre Effects. Our probing suite is drawn mostly from newswire and web text (§ 2). This is a good match for the Billion Word Benchmark (BWB) used to train the ELMo model, but a weaker match for the Books Corpus used to train the published Transformer LM. To control for this, we train a Transformer LM on the BWB, using the code and hyperparameters of Radford et al. (2018).

[4]This includes both LSTM cell weights and projection matrices between layers. Non-square matrices are orthogonal along the smaller dimension.

[5]The Transformer LM outperforms ELMo on SPR2 and Winograd coreference, but neither difference is statistically significant.

	CoVe			ELMo			Transformer LM		
Task	**Lex.**	**Full**	**Abs. Δ**	**Lex.**	**Full**	**Abs. Δ**	**Lex.**	**Full**	**Abs. Δ**
Part-of-Speech	86	94	8.4	90	**97**	6.3	88	95	6.7
Constituents	56	82	25.4	69	**85**	15.4	65	81	16.2
Dependencies	75	84	8.6	80	**94**	13.6	78	92	14.4
Entities	88	90	1.9	92	**96**	3.5	89	93	4.2
SRL (all)	60	80	20.7	74	**90**	16.0	68	86	18.4
Core roles	*56*	*81*	*24.7*	*74*	***93***	*19.0*	*65*	*88*	*22.9*
Non-core roles	*68*	*79*	*11.1*	*75*	***84***	*8.8*	*74*	*81*	*7.4*
OntoNotes coref.	73	79	6.3	75	**84**	8.7	72	84	11.9
SPR1	74	77	3.4	80	**85**	4.7	79	83	4.3
SPR2	77	80	3.6	82	83	1.0	82	**84**	1.6
Winograd coref.	**62**	52	-10.3	55	52	-3.5	57	57	0.3

Table 2: Comparison of representation models and their respective lexical baselines. Numbers reported are micro-averaged F1 score on respective test sets. **Lex.** denotes the lexical baseline (§ 4) for each model, and bold denotes the best performance on each task. Lines in *italics* are subsets of the full set of targets. SRL numbers consider core and non-core roles, but ignore references and continuations. Transformer LM is the published model trained on the Books Corpus. 95% confidence intervals (normal approximation) are approximately ± 3 for Winograd (DPR), ± 1 for SPR1 and SPR2, and ± 0.5 or smaller for all other tasks.

We find that this model performs only slightly better (+0.15 F1 on average) on our probing suite than the Books Corpus-trained model, but still underperforms ELMo by nearly 1 F1 point.

Encoding of syntactic vs. semantic information. By comparing to lexical baselines, we can measure how much the contextual information from a particular encoder improve performance on each task. Note that in all cases, the contextual representation is strictly more expressive, since it includes access to the lexical representations either by concatenation (for CoVe and the Transformer LM) or by scalar mixing weights (for ELMo).

We observe that ELMo, CoVe, and Transformer LM all follow a similar trend across our suite (Table 2), showing the largest gains on tasks which are considered to be largely syntactic, such dependency and constituent labeling, and smaller gains on tasks which are considered to require more semantic reasoning, such as SPR and Winograd. We observe small absolute improvements (+6.3 and +3.5 for ELMo) on part-of-speech tagging and entity labeling, but note that this is likely due to the strength of word-level priors on these tasks. Relative reduction in error is much higher (+66% and +44%, respectively), suggesting that ELMo does encode local type information.

Semantic role labeling benefits greatly overall, but this is predominantly due to better labeling of core roles (+19.0 F1 for ELMo) which are known to be closely tied to syntax (e.g. Punyakanok et al. (2008); Gildea & Palmer (2002)). The lexical baseline performs similarly on core and non-core roles (0.74 vs. 0.75 for ELMo), but the more semantically-oriented non-core role labels (such as purpose, cause, or negation) see only a smaller improvement from encoded context (+8.8 F1 for ELMo). The semantic proto-role labeling task (SPR1, SPR2) looks at the same type of core predicate-argument pairs but tests for higher-level semantic properties (§ 2), which we find to be only weakly captured (+1-5 F1 for ELMo) by the contextual encoder.

Effects of architecture. Focusing on the ELMo model, we ask: how much of the model's performance can be attributed to the architecture, rather than knowledge from pretraining? In Figure 2 we compare to an orthonormal encoder (§ 4) which is structurally identical to ELMo but contains no information in the recurrent weights. It can be thought of as a randomized feature function over the sentence, and provides a baseline for how the architecture itself can encode useful contextual information. We find that the orthonormal encoder improves significantly on the lexical baseline, but that overall the learned weights account for over 70% of the improvements from full ELMo.

Encoding non-local context. How much information is carried over long distances (several tokens or more) in the sentence? To estimate this, we extend our lexical baseline with a convolutional

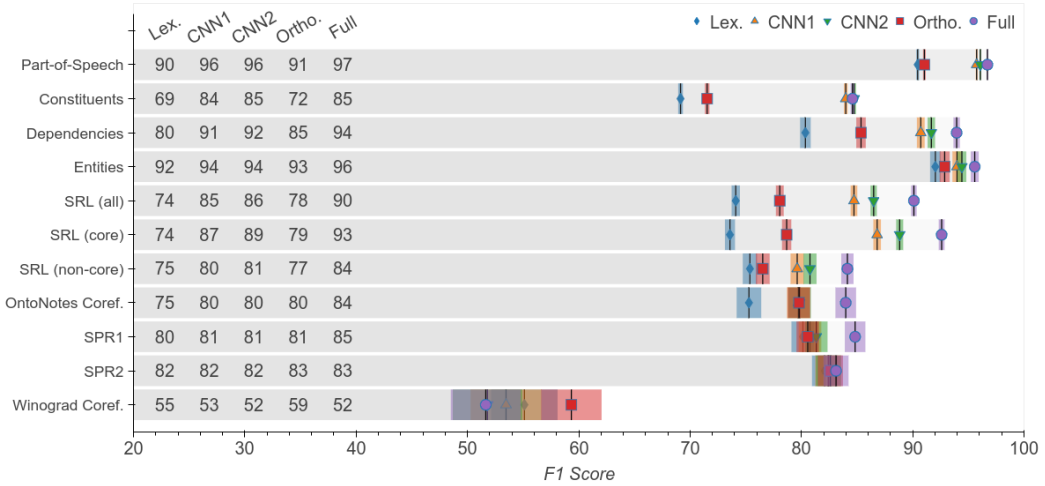

Figure 2: Additional baselines for ELMo, evaluated on the test sets. CNNk adds a convolutional layer that sees $\pm k$ tokens to each side of the center word. Lexical is the lexical baseline, equivalent to $k = 0$. Orthonormal is the full ELMo architecture with random orthonormal LSTM and projection weights, but using the pretrained lexical layer. Full (pretrained) is the full ELMo model.

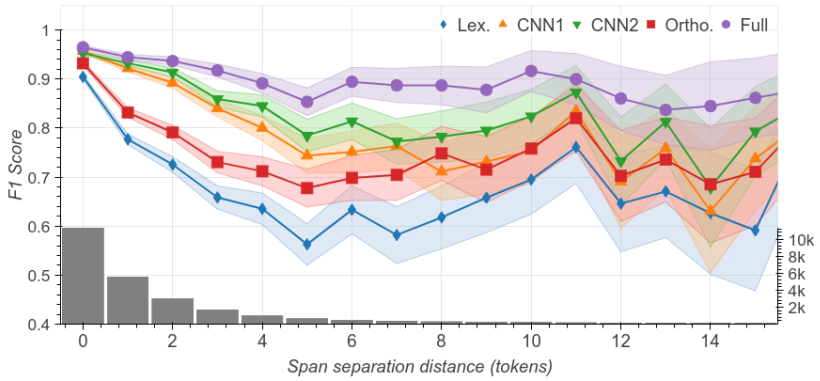

Figure 3: Dependency labeling F1 score as a function of separating distance between the two spans. Distance 0 denotes adjacent tokens. Bars on the bottom show the number of targets (in the developement set) with that distance.

layer, which allows the probing classifier to use local context. In Figure 2 we find that, averaged over each test set, adding a CNN of width 3 (± 1 token) closes 72% of the gap between the lexical baseline and full ELMo; this extends to 79% if we use a CNN of width 5 (± 2 tokens). On nonterminal constituents, we find that the CNN ± 2 model matches ELMo performance, suggesting that while the ELMo encoder propagates a large amount information about constituents (+15% F1), most of it is local in nature. We see a similar trend on the other syntactic tasks, with 80-90% of ELMo performance on dependencies, part-of-speech, and SRL core roles captured by CNN ± 2. Conversely, on more semantic tasks, such as coreference, SRL non-core roles, and SPR, the gap between full ELMo and the CNN baselines is larger. This suggests that while ELMo does not encode these phenomena as efficiently, the improvements it does bring are largely due to long-range information. We can test this hypothesis by seeing how our probing model performs with distant spans. Figure 3 shows F1 score as a function of the distance (number of tokens) between a token and its head for the dependency labeling task. The CNN models and the orthonormal encoder perform best with nearby spans, but fall off rapidly as token distance increases. The full ELMo model holds up better, with performance dropping only 7% between $d = 0$ tokens and $d = 8$, suggesting the pretrained encoder does encode useful long-distance dependencies.

6 RELATED WORK

Recent work has consistently demonstrated the strong empirical performance of contextualized word representations, including CoVe (McCann et al., 2017), ULMFit (Howard & Ruder, 2018), and ELMo (Peters et al., 2018a; Lee et al., 2018; Strubell et al., 2018; Kitaev & Klein, 2018). In response to the impressive results on downstream tasks, a line of work has emerged with the goal of understanding and comparing such pretrained representations. SentEval (Conneau & Kiela, 2018) and GLUE (Wang et al., 2018) offer suites of application-oriented benchmark tasks, such as sentiment analysis or textual entailment, which combine many types of reasoning and provide valuable aggregate metrics which are indicative of practical performance. A parallel effort, to which this work contributes, seeks to understand what is driving (or hindering) performance gains by using "probing tasks," i.e. tasks which attempt to isolate specific phenomena for the purpose of finer-grained analysis rather than application, as discussed below.

Much work has focused on probing fixed-length sentence encoders, such as InferSent (Conneau et al., 2017), specifically their ability to capture surface properties of sentences such as length, word content, and word order (Adi et al., 2016), as well as a broader set of syntactic features, such as tree depth and tense (Conneau et al., 2018). Other related work uses perplexity scores to test whether language models learn to encode properties such as subject-verb agreement (Linzen et al., 2016; Gulordava et al., 2018; Marvin & Linzen, 2018; Kuncoro et al., 2018).

Often, probing tasks take the form of "challenge sets", or test sets which are generated using templates and/or perturbations of existing test sets in order to isolate particular linguistic phenomena, e.g. compositional reasoning (Dasgupta et al., 2018; Ettinger et al., 2018). This approach is exemplified by the recently-released Diverse Natural Language Corpus (DNC) (Poliak et al., 2018), which introduces a suite of 11 tasks targeting different semantic phenomena. In the DNC, these tasks are all recast into natural language inference (NLI) format (White et al., 2017), i.e. systems must understand the targeted semantic phenomenon in order to make correct inferences about entailment.

Challenge sets which operate on full sentence encodings introduce confounds into the analysis, since sentence representation models must pool word-level representations over the entire sequence. This makes it difficult to infer whether the relevant information is encoded within the span of interest or rather inferred from diffuse information elsewhere in the sentence. One strategy to control for this is the use of minimally-differing sentence pairs (Poliak et al., 2018; Ettinger et al., 2018). An alternative approach, which we adopt in this paper, is to directly probe the token representations for word- and phrase-level properties. This approach has been used previously to show that the representations learned by neural machine translation systems encode properties like part-of-speech, semantic tags, and morphology as well as supervised models (Shi et al., 2016; Belinkov et al., 2017a;b). Blevins et al. (2018) goes further to explore how part-of-speech and hierarchical constituent structure are encoded by different pretraining objectives and at different layers of the model. Peters et al. (2018b) presents similar results for ELMo and architectural variants.

Compared to existing work, we extend sub-sentence probing to a broader range of syntactic and semantic tasks, including long-range and high-level relations such as predicate-argument structure. Our approach can incorporate existing annotated datasets without the need for templated data generation, and admits fine-grained analysis by label and by metadata such as span distance. We note that some of the tasks we explore overlap with those included in the DNC, in particular, NER, SPR and Winograd. However, our focus on probing token-level representations directly, rather than pooling over the whole sentence, provides a complementary means for analyzing these representations and diagnosing the particular advantages of contextualized vs. conventional word embeddings.

7 CONCLUSION

We introduce a suite of "edge probing" tasks designed to probe the sub-sentential structure of contextualized word embeddings. These tasks are derived from core NLP tasks and encompass a range of syntactic and semantic phenomena. We use these tasks to explore how contextual embeddings improve on their lexical (context-independent) baselines. We focus on three recently-released models for contextualized word embeddings–CoVe, ELMo, and the Transformer LM.

Based on our analysis, we find evidence suggesting the following trends. First, in general, contextualized embeddings improve over their non-contextualized counterparts largely on syntactic tasks (e.g. constituent labeling) in comparison to semantic tasks (e.g. coreference), suggesting that these embeddings encode syntax more so than higher-level semantics. Second, the performance of ELMo cannot be fully explained by a model with access to local context, suggesting that the contextualized representations do encode distant linguistic information, which can help disambiguate longer-range dependency relations and higher-level syntactic structures.

We release our data processing and model code, and hope that this can be a useful tool to facilitate understanding of, and improvements in, contextualized word embedding models.

REFERENCES

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

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. What do neural machine translation models learn about morphology? In *Proceedings of EMNLP*, 2017a.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of IJCNLP*, 2017b.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. Deep RNNs encode soft hierarchical syntax. In *Proceedings of ACL*, 2018.

Ondřej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, and Julia Kreutzer (eds.). *Proceedings of the Second Conference on Machine Translation*. 2017.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. In *Proceedings of Interspeech*, 2014.

Alexis Conneau and Douwe Kiela. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint 1803.05449*, 2018.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of EMNLP*, 2017.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single $&#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of ACL*, 2018.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J Gershman, and Noah D Goodman. Evaluating compositionality in sentence embeddings. *arXiv preprint 1802.04302*, 2018.

Allyson Ettinger, Ahmed Elgohary, Colin Phillips, and Philip Resnik. Assessing composition in sentence vector representations. In *Proceedings of COLING*, 2018.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. *arXiv preprint 1803.07640*, 2018.

Daniel Gildea and Martha Palmer. The necessity of parsing for predicate argument recognition. In *Proceedings of ACL*, 2002.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. Colorless green recurrent networks dream hierarchically. In *Proceedings of NAACL*, 2018.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of ACL*, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of ACL*, 2018.

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

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

Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. In *Proceedings of ACL*, July 2018.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of ACL*, 2018.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end neural coreference resolution. In *Proceedings of EMNLP*, 2017.

Kenton Lee, Luheng He, and Luke Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of NAACL*, 2018.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2012.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the ACL*, 4(1):521–535, 2016.

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

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Proceedings of NIPS*, 2017.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, 2013.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, et al. Universal dependencies 1.2. 2015.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Proceedings of NIPS*, 2017.

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

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

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. Dissecting contextual word embeddings: Architecture and representation. *arXiv preprint 1808.08949*, 2018b.

Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of EMNLP*, 2018.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287, 2008.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *https://blog.openai.com/language-unsupervised*, 2018.

Altaf Rahman and Vincent Ng. Resolving complex cases of definite pronouns: The Winograd schema challenge. In *Proceedings of EMNLP*, 2012.

Rachel Rudinger, Adam Teichert, Ryan Culkin, Sheng Zhang, and Benjamin Van Durme. Neural Davidsonian semantic proto-role labeling. In *Proceedings of EMNLP*, 2018.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of ACL*, 2016.

Xing Shi, Inkit Padhi, and Kevin Knight. Does string-based neural MT learn source syntax? In *Proceedings of EMNLP*, 2016.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of EMNLP*, 2018.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, 2014.

Adam Teichert, Adam Poliak, Benjamin Van Durme, and Matthew Gormley. Semantic proto-role labeling. In *Proceedings of AAAI*, 2017.

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

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint 1804.07461*, 2018.

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

Aaron Steven White, Pushpendre Rastogi, Kevin Duh, and Benjamin Van Durme. Inference is everything: Recasting semantic resources into a unified evaluation framework. In *Proceedings of IJCNLP*, 2017.

Kelly W. Zhang and Samuel R. Bowman. Language modeling teaches you more syntax than translation does: Lessons learned through auxiliary task analysis. *arXiv preprint 1809.10040*, 2018.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of ICCV*, 2015.

A DATASET STATISTICS

Table 3: For each probing task, corpus summary statistics of the number of labels, examples, tokens and targets (split by train/dev/test). Examples generally refer to sentences. For semantic role labeling, they instead refer to the total number of frames.

Task	$\lvert\mathcal{L}\rvert$	**Examples**	**Tokens**	**Total Targets**
Part-of-Speech labeling	48	116K / 16K / 12K	2.2M / 305K / 230K	2.1M / 290K / 212K
Constituent labeling	30	116K / 16K / 12K	2.2M / 305K / 230K	1.9M / 255K / 191K
Dependency labeling	49	13K / 2.0K / 2.1K	204K / 25K / 25K	204K / 25K / 25K
Named entity labeling	18	116K / 16K / 12K	2.2M / 305K / 230K	128K / 20K / 13K
Semantic role labeling	66	253K / 35K / 24K	6.6M / 934K / 640K	599K / 83K / 56K
Core roles	6	253K / 35K / 24K	6.6M / 934K / 640K	411K / 57K / 38K
Non-core roles	21	253K / 35K / 24K	6.6M / 934K / 640K	170K / 24K / 16K
OntoNotes coreference	2	116K / 16K / 12K	2.2M / 305K / 230K	248K / 43K / 40K
Semantic proto-roles 1	18	3.8K / 513 / 551	81K / 11K / 12K	7.6K / 1.1k / 1.1K
Semantic proto-roles 2	20	2.2K / 291 / 276	47K / 4.9K / 5.6K	4.9K / 630 / 582
Winograd coreference	2	1.0K / 2.0K / 2.1K	14K / 8.0K / 14K	1.8K / 949 / 379

B MODEL DETAILS

Because the vectors have varying dimension across probed models, and to improve performance we first project the vectors down to 256 dimensions:

$$e_i^{(k)} = A^{(k)} e_i + b^{(k)} \tag{1}$$

We use separate projections ($k = 1, 2$) so that the model can extract different information from $s^{(1)}$ (for example, a predicate) and $s^{(2)}$ (for example, an argument). We then apply a pooling operator over the representations within a span to yield a fixed-length representation:

$$r^{(k)}(s_k) = r^{(k)}(i_k, j_k) = \mathrm{Pool}(e_{i_k}^{(k)}, e_{i_k+1}^{(k)}, \ldots, e_{j_k-1}^{(k)}) \tag{2}$$

We use the self-attentional pooling operator from Lee et al. (2017) and He et al. (2018). This learns a weight $z_i^{(k)} = W_{att}^{(k)} e_i^{(k)}$ for each token, then represents the span as a sum of the vectors $e_{i_k}^{(k)}, e_{i_k+1}^{(k)}, \ldots, e_{j_k-1}^{(k)}$ weighted by $a_i^{(k)} = \mathrm{softmax}(\mathbf{z}^{(k)})_i$.

Finally, the pooled span representations are fed into a two-layer MLP followed by a sigmoid output layer:

$$\begin{gathered} h = MLP([r^{(1)}(s^{(1)}), r^{(2)}(s^{(2)})]) \\ P(\mathrm{label}_\ell = 1) = \sigma(Wh = b)_\ell \quad \text{for} \quad \ell = 0, \ldots, |\mathcal{L}| \end{gathered} \tag{3}$$

We train by minimizing binary cross entropy against the set of true labels. While convention on many tasks (e.g. SRL) is to use a softmax loss, this enforces an exclusivity constraint. By using a per-label sigmoid our model can estimate each label independently, which allows us to stratify our analysis (see § 5) to individual labels or groups of labels within a task.

With the exception of ELMo scalars, we hold the weights of the sentence encoder (§ 3.2) fixed while we train our probing classifier. We train using the Adam optimizer (Kingma & Ba, 2015) with a batch size[6] of 32, an initial learning rate of 1e-4, and gradient clipping with max L_2 norm

[6]For most tasks this is $b = 32$ sentences, which each have a variable number of target spans. For SRL, this is $b = 32$ predicates.

of 5.0. We evaluate on the validation set every 1000 steps (or every 100 for SPR1, SPR2, and Winograd), halve the learning rate if no improvement is seen in 5 validations, and stop training if no improvement is seen in 20 validations.

C CONTEXTUAL REPRESENTATION MODELS

CoVe The CoVe model (McCann et al., 2017) is a two-layer biLSTM trained as the encoder side of a sequence-to-sequence(Sutskever et al., 2014) English-German machine translation model. We use the original authors' implementation and the best released pre-trained model [7]. This model is trained on the WMT2017 dataset Bojar et al. (2017) which contains approximately 7 million sentences of English text. Following McCann et al. (2017), we concatenate the activations of the top-layer forward and backward LSTMs ($d = 300$ each) with the pre-trained GloVe (Pennington et al., 2014) embedding[8] ($d = 300$) of each token, for a total representation dimension of $d = 900$.

ELMo The ELMo model (Peters et al., 2018a) is a two layer LSTM trained as the concatenation of a forward and a backward language model, and built over a context-independent character CNN layer. We use the original authors' implementation as provided in the AllenNLP (Gardner et al., 2018) toolkit[9] and the standard pre-trained model trained on the Billion Word Benchmark (BWB) (Chelba et al., 2014)We take the (fixed, contextual) representation of token i to be the set of three vectors $h_{0,i}$, $h_{1,i}$, and $h_{2,i}$ containing the activations of each layer of the ELMo model. Following Equation 1 of Peters et al. (2018a), we learn task-specific scalar parameters and take a weighted sum:

$$e_i = \gamma \left(s_0 h_{0,i} + s_1 h_{1,i} + s_2 h_{2,i} \right) \quad \text{for } i = 0, 1, \ldots, n \tag{4}$$

to give 1024-dimensional representations for each token.

Transformer LM The Transformer LM of Radford et al. (2018) was recently shown to outperform ELMo on a number of downstream tasks, and as of submission holds the highest score on the GLUE benchmark (Wang et al., 2018).It consists of a 12-layer Transformer (Vaswani et al., 2017) model, trained as a left-to-right language model using masked attention. We use a PyTorch reimplementation of the model [10], and the pre-trained weights[11] trained on the Toronto Book Corpus (Zhu et al., 2015)Unlike Radford et al. (2018), we hold the Transformer weights fixed while training our probing model in order to better understand what information is available from the pre-training procedure alone. Furthermore, to facilitate more direct comparison with ELMo and CoVe we concatenate the activations of the final Transformer layer ($d = 768$) with the context-independent subword embeddings ($d = 768$) to give contextual vectors of $d = 1536$ for each (sub)-token.

D RETOKENIZATION

The pre-trained encoder models expect a particular tokenization of the input string, which does not always match the original tokenization of each probing set. To correct this we retokenize the probing data to match the tokenization of each encoder, which for CoVe is Moses tokenization, and for the Transformer LM is a custom byte-pair-encoding (BPE) model (Sennrich et al., 2016). We then align the spans to the new tokenization using a heuristic projection based on byte-level Levenshtein distance.

The source data for our probing tasks is annotated with respect to a particular tokenization, typically the conventions of the source treebanks (Penn Treebank, Universal Dependencies, and OntoNotes 5.0). This does not always align to the tokenization of the pre-trained representation models. Consider a dummy sentence:[12]

[7] https://github.com/salesforce/cove

[8] `glove.840b.300d` from https://nlp.stanford.edu/projects/glove/

[9] https://allennlp.org/ version 0.5.1

[10] https://github.com/huggingface/pytorch-openai-transformer-lm, which we manually verified to produce identical activations to the authors' TensorFlow implementation.

[11] https://github.com/openai/finetune-transformer-lm

[12] The authors do, in fact, like pineapples, but this sentence has an apostrophe.

- Text: `I don't like pineapples.`
- Native: `[I do n't like pineapples .]`
- Moses: `[I do n \'t like pineapples .]`
- BPE: `[_i _do _n't _like _pinea pples .]`

An annotation on the word "pineapples" might be expressed as $s = [4, 5)$ in the original ("native") tokenization, but the corresponding text is span $s_{\text{Moses}} = [5, 6)$ under Moses tokenization and $s_{\text{BPE}} = [5, 7)$ under the particular BPE model above.

We resolve this by aligning the source and target tokenization using Levenshtein distance. We take the source tokenization $[s_0, s_1, \ldots, s_m]$ as given, and treat the target tokenizer as a black-box function from a string $\tilde{S}$ to a list of tokens $[t_0, t_1, \ldots, t_n]$ (note that in general, $n \neq m$). Let $\tilde{S}$ be the source string. We create a target string $\tilde{T}$ by joining $[t_0, t_1, \ldots, t_n]$ with spaces, and then compute a byte-level Levenshtein alignment[13] $\tilde{A} = \text{Align}(\tilde{T}, \tilde{S})$. We then compute token-to-byte alignments $U = \text{Align}([t_0, t_1, \ldots, t_n], \tilde{T})$ and $V = \text{Align}([s_0, s_1, \ldots, s_m], \tilde{S})$. Representing the alignments as boolean adjacency matricies, we can compose them to form a token-to-token alignment $A = U\tilde{A}V^T$.

We then represent each source span as a boolean vector with 1s inside the span and 0s outside, e.g. $[2, 4) = [0, 0, 1, 1, 0, 0, \ldots] \in \{0, 1\}^m$, and project through the alignment A to the target side. We recover a target-side span from the minimum and maximum nonzero indices.

[13] We use the `python-Levenshtein` package, https://pypi.org/project/python-Levenshtein/