Putting words in context: LSTM language models and lexical ambiguity

Laura Aina **Kristina Gulordava** **Gemma Boleda**
Universitat Pompeu Fabra
Barcelona, Spain
{firstname.lastname}@upf.edu

Abstract

In neural network models of language, words are commonly represented using context-invariant representations (word embeddings) which are then put in context in the hidden layers. Since words are often ambiguous, representing the contextually relevant information is not trivial. We investigate how an LSTM language model deals with lexical ambiguity in English, designing a method to probe its hidden representations for lexical and contextual information about words. We find that both types of information are represented to a large extent, but also that there is room for improvement for contextual information.

1 Introduction

In language, a word can contribute a very different meaning to an utterance depending on the context, a phenomenon known as lexical ambiguity (Cruse, 1986; Small et al., 2013). This variation is pervasive and involves both morphosyntactic and semantic aspects. For instance, in the examples in Table 1, *show* is used as a verb in Ex. (1), and as a noun in Ex. (2-3), in a paradigmatic case of morphosyntactic ambiguity in English. Instead, the difference between Ex. (2) and (3) is semantic in nature, with *show* denoting a TV program and an exhibition, respectively. Semantic ambiguity covers a broad spectrum of phenomena, ranging from quite distinct word senses (e.g. *mouse* as animal or computer device) to more subtle lexical modulation (e.g. *visit a city / an aunt / a doctor*; Cruse, 1986). This paper investigates how deep learning models of language, and in particular Long Short-Term Memory Networks (LSTMs) trained on Language Modeling, deal with lexical ambiguity.[1]

In neural network models of language, words in a sentence are commonly represented through word-level representations that do not change across contexts, that is, "static" word embeddings. These are then passed to further processing layers, such as the hidden layers in a recurrent neural network (RNN). Akin to classic distributional semantics (Erk, 2012), word embeddings are formed as an abstraction over the various uses of words in the training data. For this reason, they are apt to represent context-invariant information about a word —its **lexical** information— but not the contribution of a word in a particular context —its **contextual** information (Erk, 2010). Indeed, word embeddings subsume information relative to various senses of a word (e.g., *mouse* is close to words from both the animal and computer domain; Camacho-Collados and Pilehvar, 2018).

Classic distributional semantics attempted to do composition to account for contextual effects, but it was in general unable to go beyond short phrases (Baroni, 2013); newer-generation neural network models have supposed a big step forward, as they can natively do composition (Westera and Boleda, 2019). In particular, the hidden layer activations in an RNN can be seen as *putting words in context*, as they combine the word embedding with information coming from the context (the adjacent hidden states). The empirical success of RNN models, and in particular LSTM architectures, at fundamental tasks like Language Modeling (Jozefowicz et al., 2015) suggests that they are indeed capturing relevant contextual properties. Moreover, contextualized representations derived from such models have been shown to be very informative as input for lexical disambiguation tasks (e.g. Melamud et al., 2016; Peters et al., 2018).

We here present a method to probe the extent to which the hidden layers of an LSTM language trained on English data represent lexical and contextual information about words, in order to investigate how the model copes with lexical ambiguity.

[1]Code at: https://github.com/amore-upf/LSTM_ambiguity

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3342–3348
Florence, Italy, July 28 - August 2, 2019.

Examples	LexSub	**w** NN	**s** NN	**w**&**s** NN
(1) . . . *I clapped her shoulder to* ***show*** *I was not laughing at her. . .*	demonstrate, display, indicate, prove, clarify	demonstrate, exhibit, indicate, offer, reveal	indicate, demonstrate, suggest, prove,	indicate, demonstrate, prove, ensure, suggest
(2) . . . *The* ***show*** *[. . .] revolutionized the way America cooks and eats. . .*	program, series, broadcast, presentation	demonstrate, exhibit, indicate, offer, reveal	series, program, production, miniseries, trilogy	series, program, production, broadcast
(3) . . . *The inauguration of Dubai Internet City coincides with the opening of an annual IT* ***show*** *in Dubai. . . .*	exhibition, conference, convention, demonstration	demonstrate, exhibit, indicate, offer, reveal	conference, event, convention, symposium, exhibition	conference, event, exhibition, symposium, convention

Table 1: Examples from the LexSub dataset (Kremer et al., 2014) and nearest neighbors for target representations.

Our work follows a recent strand of research that purport to identify what linguistic properties deep learning models are able to capture (Linzen et al., 2016; Adi et al., 2017; Gulordava et al., 2018; Conneau et al., 2018; Hupkes et al., 2018, a.o.). We train diagnostic models on the tasks of retrieving the embedding of a word and a representation of its contextual meaning, respectively —the latter obtained from a Lexical Substitution dataset (Kremer et al., 2014). Our results suggest that LSTM language models heavily rely on the lexical information in the word embeddings, at the expense of contextually relevant information. Although further analysis is necessary, this suggests that there is still much room for improvement to account for contextual meanings. Finally, we show that the hidden states used to predict a word – as opposed to those that receive it as input – display a bias towards contextual information.

2 Method

Language model. As our base model, we employ a word-level bidirectional LSTM (Schuster and Paliwal, 1997; Hochreiter and Schmidhuber, 1997) language model (henceforth, LM) with three hidden layers. Each input word at timestep t is represented through its word embedding $\mathbf{w}_t$; this is fed to both a forward and a backward stacked LSTMs, which process the sequence left-to-right and right-to-left, respectively (Eqs. (1-2) describe the forward LSTM). To predict the word at t, we obtain output weights by summing the activations of the last hidden layers of the forward and backward LSTMs at timesteps $t-1$ and $t+1$, respectively, and applying a linear transformation followed by softmax (Eq. 3, where L is the number of hidden layers). Thus, a word is predicted using both its left and right context jointly, akin to the *context2vec* architecture (Melamud et al., 2016) but differently from, e.g., the BiLSTM architecture used for ELMo (Peters et al., 2018).

$$\mathbf{h}_t^1 = \text{LSTM}^1(\mathbf{w}_t, \mathbf{h}_{t-1}^1) \quad (1)$$

$$\mathbf{h}_t^i = \text{LSTM}^i(\mathbf{h}_t^{i-1}, \mathbf{h}_{t-1}^i) \quad (2)$$

$$\mathbf{o}_t = softmax(f(\overrightarrow{\mathbf{h}}_{t-1}^L + \overleftarrow{\mathbf{h}}_{t+1}^L)) \quad (3)$$

We train the LM on the concatenation of English text data from a Wikipedia dump[2], the British National Corpus (Leech, 1992), and the UkWaC corpus (Ferraresi et al., 2008).[3] More details about the training setup are specified in Appendix A.1. The model achieves satisfying performances on test data (perplexity: 18.06).

For our analyses, we deploy the trained LM on a text sequence and extract the following activations of each hidden layer; Eq. (4) and Fig. 1.

$$\{\overrightarrow{\mathbf{h}}_t^i | i \leq L\} \cup \{\overleftarrow{\mathbf{h}}_t^i | i \leq L\} \quad (4)$$

$$\mathbf{h}_t^i = [\overrightarrow{\mathbf{h}}_t^i; \overleftarrow{\mathbf{h}}_t^i] \quad (5)$$

$$\mathbf{h}_{t\pm1}^i = [\overrightarrow{\mathbf{h}}_{t-1}^i; \overleftarrow{\mathbf{h}}_{t+1}^i] \quad (6)$$

At timestep t, for each layer, we concatenate the forward and backward hidden states; Eq. (5). We refer to these vectors as **current hidden states**. As they are obtained processing the word at t as input and combining it with information from the context, we can expect them to capture the relevant contribution of such word (e.g., in Fig. 1 the mouse-as-animal sense). As a comparison, we also extract activations obtained by processing the text sequence up to $t-1$ and $t+1$ in the forward and backward LSTM, respectively, hence excluding the word at t. We concatenate the forward and backward states of each layer; Eq. (6). While these

[2]From 2018/01/03, https://dumps.wikimedia.org/enwiki/

[3]50M tokens from each corpus, in total 150M (train/valid/test: 80/10/10%); vocabulary size: 50K.

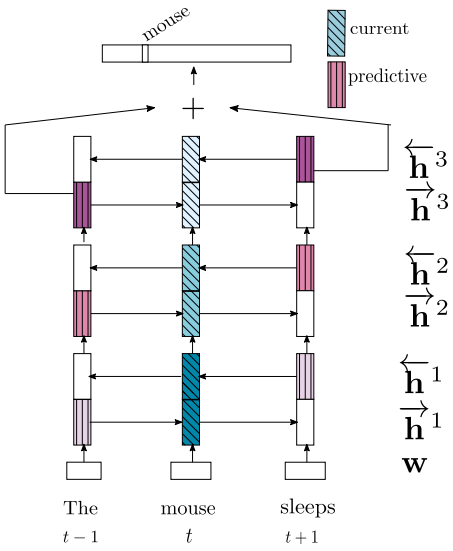

Figure 1: Language model and extracted representations. The different shades across layers reflect the different performances in the probe tasks (darker = higher)

activations do not receive the word at t as input, they are relevant because they are used to predict that word as output. We refer to them as **predictive hidden states**. These may capture some aspects of the word (e.g., in Fig. 1, that it is a noun and denotes something animate), but are likely to be less accurate than the current states, since they do not observe the actual word.

Probe tasks. We aim to assess to what extent the hidden states in the LM carry over the lexical and context-invariant information in the input word embedding, and how much they instead represent the contextual meaning of the word. To this end, we rely on vector representations of lexical and contextual word information. As for the former, we can directly use the word embeddings of the LM ($\mathbf{w}$); it is instead more challenging to find a representation of the contextual meaning.

Our solution is to use Lexical Substitution data (McCarthy and Navigli, 2009) and, in particular, the large dataset by Kremer et al., 2014 (henceforth, LexSub; see Table 1). In this dataset, words in context (up to 3 sentences) are annotated with a set of paraphrases given by human subjects. Since contextual substitutes reflect differences among uses of a word (for instance, *demonstrate* paraphrases *show* in a context like Ex. (1), but not in Ex. (2)), this type of data is often used as an evaluation benchmark for contextual representations of words (e.g., Erk and Padó, 2008; Melamud et al., 2016; Garí Soler et al., 2019). We leverage LexSub to build proxies for ground-truth representations of the contextual meaning of words. We define two types of representations, inspired by previous work that proposed simple vector operations to combine word representations (Mitchell and Lapata, 2010; Thater et al., 2011, a.o.): the average embedding of the substitute words (henceforth, $\mathbf{s}$), and the average embedding of the union of the substitute words and the target word ($\mathbf{w\&s}$). As Table 1 qualitatively shows, the resulting representations tend to be close to the substitute words and reflect the contextual nuance conveyed by the word; in the case of $\mathbf{w\&s}$, they also retain a strong similarity to the embedding of the target word.[4]

We frame our analyses as supervised probe tasks: a diagnostic model learns to "retrieve" word representations out of the hidden states; the rate of success of the model is taken to measure the amount of information relevant to the task that its input contains. Given current or predictive states as inputs, we define three diagnostic tasks:

- WORD: predict $\mathbf{w}$
- SUB: predict $\mathbf{s}$
- WORD&SUB: predict $\mathbf{w\&s}$

The WORD task is related to the probe tasks introduced in Adi et al. (2017) and Conneau et al. (2018), which, given a hidden state, require to predict the words that a sentence encoder has processed as input. Note that, while these authors predict words by their discrete index, we are predicting the complete multi-dimensional embedding of the word. Our test quantifies not only whether the model is tracking the identity of the input word, but also how much of its information it retains.

We train distinct probe models for each task and type of input ($\mathbf{i}$; e.g., current hidden state at layer 1). A model consists of a non-linear transformation from an input vector $\mathbf{i}$ (extracted from the LM) to a vector with the dimensionality of the word embeddings (Eq. 7, where $\hat{\mathbf{r}}$ is one of $\hat{\mathbf{w}}$, $\hat{\mathbf{s}}$, $\hat{\mathbf{w\&s}}$ for WORD, SUB, and WORD&SUB tasks, respectively). The models are trained through max-margin loss, optimizing the cosine similarity between $\hat{\mathbf{r}}$ and the target representation against the similarities between $\hat{\mathbf{r}}$ and 5 negative samples (details in Appendix A.2).

$$\hat{\mathbf{r}} = tanh(W\ \mathbf{i} + b) \quad (7)$$

[4]These vectors are close to the related word embedding (0.45 and 0.66 mean cosine, see Table 2, row $\mathbf{w}_t$), but also different from it: on average, $\mathbf{s}$ and $\mathbf{w\&s}$ share 17 and 25% of the top-10 neighbors with $\mathbf{w}$, respectively (statistics from training data, excluding the word itself from neighbors).

input	WORD	SUB	WORD&SUB
$\mathbf{w}_t$	1	.45 (±.14)	.66 (±.09)
avg_{ctxt}	.35 (±.10)	.16 (±.11)	.24 (±.12)
$\mathbf{h}_t^1$	**.84** (±.2)	**.61** (±.14)	**.71** (±.11)
$\mathbf{h}_t^2$	.74 (±.12)	.60 (±.13)	.69 (±.11)
$\mathbf{h}_t^3$	.64 (±.12)	.58 (±.13)	.65 (±.11)
$\mathbf{h}_{t\pm1}^1$	.25 (±.16)	.36 (±.16)	.38 (±.16)
$\mathbf{h}_{t\pm1}^2$	.27 (±.16)	.39 (±.16)	.41 (±.16)
$\mathbf{h}_{t\pm1}^3$	**.29** (±.15)	**.41** (±.16)	**.43** (±.16)

Table 2: Results of probe tasks for current ($\mathbf{h}_t^i$) and predictive ($\mathbf{h}_{t\pm1}^i$) hidden states.

We adapt the LexSub data to our setup as follows. Since substitutes are provided in their lemmatized form, we only consider datapoints where the word form is identical to the lemma so as to exclude effects due to morphosyntax (e.g., asking the models to recover *play* when they observe *played*).[5] We require that at least 5 substitutes per datapoint are in the LM vocabulary to ensure quality in the target representations. LexSub data come with a validation/test split; since we need training data, we create a new random partitioning into train/valid/test (70/10/20%, with no overlapping contexts among splits). The final data consist of 4.7K/0.7K/1.3K datapoints for train/valid/test.

3 Results

The results of the probe tasks on test data are presented in Table 2. We report the mean and standard deviation of the **cosine** similarity between the output representations ($\hat{\mathbf{w}}$, $\hat{\mathbf{s}}$, $\hat{\mathbf{w\&s}}$) and the target ones ($\mathbf{w}$, $\mathbf{s}$, $\mathbf{w\&s}$). This evaluates the degree to which the word representations can be retrieved from the hidden states. For comparison, we also report the cosine scores between the targets and two baseline representations: the word embedding itself and the average of word embeddings of a 10-word window around the target word (avg_{ctxt}).[6] Overall, the models do better than these unsupervised baselines, with exceptions.[7]

Current hidden states. Both lexical and contextual representations can be retrieved from the current hidden states ($\mathbf{h}_t^i$) to a large extent (cosines

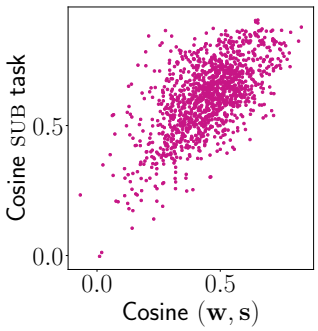

Figure 2: Similarity of lexical and contextual vector ($\mathbf{w}$ - $\mathbf{s}$) vs. similarity of target and prediction in SUB for $\mathbf{h}_t^1$.

.58-.84), but retrieving the former is much easier than the latter (.64-.84 vs. .58-71). This suggests that the information in the word embedding is better represented in the hidden states than the contextually relevant one. In all three tasks, performance degrades closer to the output layer (from $\mathbf{h}_t^1$ to $\mathbf{h}_t^3$), but the effect is more pronounced for the WORD task (84/.74/.64). Word embeddings are part of the input to the hidden state, and the transformation learned for this task can be seen as a decoder in an auto-encoder, reconstructing the original input; the further the hidden layer is from the input, the more complex the function is to reverse-engineer. Crucially, the high performance at reconstructing the word embedding suggests that lexical information is retained in the hidden layers, possibly including also contextually irrelevant information (e.g., in Ex. (4) in Table 3 $\hat{\mathbf{w}}$ is close to verbs, even if *share* is here a noun).

Contextual information ($\mathbf{s}$ and $\mathbf{w\&s}$) seems to be more stable across processing layers, although overall less present (cf. lower results). Table 3 reports one example where the learned model displays relevant contextual aspects (Ex. (4), *share*) and one where it does not (Ex. (5), *studio*). Qualitative analysis shows that morphosyntactic ambiguity (e.g., *share* as a noun vs. verb) is more easily discriminated, while semantic distinctions pose more challenges (e.g., *studio* as a room vs. company). This is not surprising, since the former tends to correlate with clearer contextual cues. Furthermore, we find that the more the contextual representation is aligned to the lexical one, the easier it is to retrieve the former from the hidden states (e.g., correlation $\cos(\mathbf{w},\mathbf{s})$ - $\cos(\hat{\mathbf{s}},\mathbf{s})$, for $\mathbf{h}_t^1$: Pearson's $\rho = .62^{***}$; Fig. 2): that is, it is harder to resolve lexical ambiguity when the contextual meaning is less represented in the word embedding (e.g., less frequent uses). This suggests that the LM heavily relies on the informa-

[5] We also exclude substitutes that are multi-word expressions and the datapoints involving words that are part of a compound (e.g., *fast* in *fast-growing*).

[6] We exclude out-of-vocabulary words and punctuation.

[7] The first cell is 1 as it involves the same representation.

Context	LexSub	WORD: $\hat{\mathbf{w}}$ NN	SUB: $\hat{\mathbf{s}}$ NN	WORD&SUB: $\mathbf{w}\hat{\&}\mathbf{s}$ NN
(4) ... *The financial-services company will pay 0.82 share for each Williams* ***share*** *...*	stock, dividend, interest, stake, unit	stake, owe, discuss, coincide, reside	portion, amount, percentage, fraction	stake, percentage, portion, spend, proportion
(5) ... *Sony's effort to hire producers Jon Peters and Peter Guber to run the* ***studio****...*	business, company, facility, film, lot	lab, troupe, classroom, apartment, booth	room, gallery, troupe, journal, house	room, troupe, lab, audience, department
(6) ... *I had [...] told her that we needed other* ***company*** *than our own ...*	friend, acquaintance, visitor, accompaniment, associate	retailer, trader, firm, maker, supplier	firm, corporation, organisation, conglomerate, retailer	corporation, firm, conglomerate, retailer, organisation

Table 3: Examples with nearest neighbours of the representations predicted in the first current hidden layer.

tion in the word embedding, making it challenging to diverge from it when contextually relevant (see Ex. (6) in Table 3).

Current vs. predictive hidden states. The predictive hidden states are obtained without observing the target word; hence, recovering word information is considerably harder than for current states. Indeed, we observe worse results in this condition (e.g., below avg_{ctxt} in the WORD task); we also observe two patterns that are opposite to those observed for current states, which shed light on how LSTM LMs track word information.

For predictive states, results improve closer to the output (from layer 1 to 3; they instead degrade for current states). We link this to the double objective that a LM has when it comes to word information: to integrate a word passed as input, and to predict one as output. Our results suggest that the hidden states keep track of information for both words, but lower layers focus more on the processing of the input and higher ones on the predictive aspect (see Fig. 1). This is in line with previous work showing that activations close to the output tend to be task-specific (Liu et al., 2019).

Moreover, from predictive states, it is easier to retrieve contextual than lexical representations (.41/.43 vs. .29; the opposite was true for current states). Our hypothesis is that this is due to a combination of two factors. On the one hand, predictive states are based solely on contextual information, which highlights only certain aspects of a word; for instance, the context of Ex. (2) in Table 1 clearly signals that a noun is expected, and the predictive states in a LM should be sensitive to this kind of cue, as it affects the probability distribution over words. On the other hand, lexical representations are underspecified; for instance, the word embedding for *show* abstracts over both verbal and nominal uses of the word. Thus, it makes sense that the predictive state does not capture contextually irrelevant aspects of the word embedding, unlike the current state (note however that, as stated above, the overall performance of the current state is better, because it has access to the word actually produced).

4 Future work

We introduced a method to study how deep learning models of language deal with lexical ambiguity. Though we focused on LSTM LMs for English, this method can be applied to other architectures, objective tasks, and languages; possibilities to explore in future work. We also plan to carry out further analyses aimed at individuating factors that challenge the resolution of lexical ambiguity (e.g., morphosyntactic vs. semantic ambiguity, frequency of a word or sense, figurative uses), as well as clarifying the interaction between prediction and processing of words within neural LMs.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 715154), and from the Ramón y Cajal programme (grant RYC-2015-18907). We gratefully acknowledge the support of NVIDIA Corporation with the donation of GPUs used for this research, and the computer resources at CTE-POWER and the technical support provided by Barcelona Supercomputing Center (RES-FI-2018-3-0034). This paper reflects the authors' view only, and the EU is not responsible for any use that may be made of the information it contains.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *Proceedings of 5th ICLR International Conference on Learning Representations*.

Marco Baroni. 2013. Composition in distributional semantics. *Language and Linguistics Compass*, 7(10):511–522.

Jose Camacho-Collados and Taher Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence*, 63(1):743–788.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2126–2136.

Alan Cruse. 1986. *Lexical semantics*. Cambridge University Press.

Katrin Erk. 2010. What is word meaning, really? (and how can distributional models help us describe it?). In *Proceedings of the 2010 Workshop on Geometrical Models of Natural Language Semantics*, pages 17–26.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the EMNLP Conference on Empirical Methods in Natural Language Processing*, pages 897–906.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukWac, a very large web-derived corpus of English. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54.

Aina Garí Soler, Anne Cocos, Marianna Apidianaki, and Chris Callison-Burch. 2019. A comparison of context-sensitive models for lexical substitution. In *Proceedings of the 13th International Conference on Computational Semantics (IWCS)*.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 NAACL-HLT Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1195–1205.

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

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What substitutes tell us-analysis of an" all-words" lexical substitution corpus. In *Proceedings of the 14th EACL Conference of the European Chapter of the Association for Computational Linguistics*, pages 540–549.

Geoffrey Neil Leech. 1992. 100 million words of English: the British National corpus (BNC).

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

Nelson F Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 NAACL-HLT Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Diana McCarthy and Roberto Navigli. 2009. The English lexical substitution task. *Language resources and evaluation*, 43(2):139–159.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.

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

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 NAACL-HLT Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Steven L Small, Garrison W Cottrell, and Michael K Tanenhaus. 2013. *Lexical Ambiguity Resolution: Perspective from Psycholinguistics, Neuropsychology and Artificial Intelligence*. Elsevier.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word meaning in context: A simple and effective vector model. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1134–1143.

Matthijs Westera and Gemma Boleda. 2019. Don't blame distributional semantics if it can't do entailment. In *Proceedings of the 13th International Conference on Computational Semantics (IWCS)*, pages 120–133.

A Appendix

A.1 Language model

The hidden layers are of sizes 600/600/300 respectively, while the word embeddings are of size 300. The language model was trained optimizing the log-likelihood of a target word given its surrounding context, with stochastic gradient descent for 20 epochs with decaying learning rate using Adam optimiser (Kingma and Ba, 2014). The initial learning rate was 0.0005 for batch size of 32. Dropout was set to 0.2 and applied to the input embedding, and the outputs of the LSTM layers. At training time, the text data is fed to the model in sequences of 100 tokens.

A.2 Diagnostic models

We train separate models for each combination of task and input type. Each model consist of a linear transformation and a $tahn$ non-linearity, trained using Cosine Embedding Loss (PyTorch 0.4, Paszke et al., 2017) and Adam optimiser, with early stopping based on validation loss. We carried out hyperparameter search based on validation loss for each of the model types in order to set batch size and initial learning rate. We report the final settings for each combination of input and task in Table 4.

At training time, for each positive target word, we obtain 5 negative targets by sampling words from the frequency quartile of the postive target (frequency is computed on the training corpus of the language model). We always exclude the target word, as well as the substitute words in the SUB and WORD&SUB conditions, from the negative samples. Given the input vector, we maximize the margin of the resulting output vector $\hat{r}$ to the embeddings of the negative samples ($i = -1$), and minimize the distance of the output vector to the target representation of the positive instance ($i = 1$; Eq. 8).

input	WORD	SUB	WORD&SUB
$\mathbf{h}_t^1$	$16, 5 \times 10^{-5}$	$32, 1 \times 10^{-4}$	$32, 5 \times 10^{-5}$
$\mathbf{h}_t^2$	$16, 5 \times 10^{-5}$	$64, 5 \times 10^{-4}$	$64, 5 \times 10^{-4}$
$\mathbf{h}_t^3$	$16, 5 \times 10^{-5}$	$128, 5 \times 10^{-4}$	$16, 5 \times 10^{-5}$
$\mathbf{h}_{t\pm1}^1$	$128, 1 \times 10^{-3}$	$128, 1 \times 10^{-3}$	$128, 5 \times 10^{-4}$
$\mathbf{h}_{t\pm1}^2$	$16, 1 \times 10^{-4}$	$64, 5 \times 10^{-4}$	$16, 5 \times 10^{-4}$
$\mathbf{h}_{t\pm1}^3$	$128, 1 \times 10^{-3}$	$16, 1 \times 10^{-4}$	$128, 5 \times 10^{-4}$

Table 4: Hyperparameter settings in the diagnostic models (batch size, initial learning rate)

$$L(\hat{r}, r, i) = \begin{cases} 1 - cos(\hat{r}, r) & \text{if } i = 1 \\ \\ max(0, cos(\hat{r}, r) - \text{margin}) & \text{if } i = -1 \end{cases} \quad (8)$$

At each training epoch, new negative instances are sampled, and the data is shuffled.