Multi-Sense Language Modelling

Lekkas, Andrea; Schneider-Kamp, Peter; Augenstein, Isabelle

Published in:
arXiv

DOI:
https://arxiv.org/abs/2012.05776

Publication date:
2020

Document version
Publisher's PDF, also known as Version of record

Citation for published version (APA):
Lekkas, A., Schneider-Kamp, P., & Augenstein, I. (2020). Multi-Sense Language Modelling. *arXiv*, *CyRR 2020*.
https://doi.org/https://arxiv.org/abs/2012.05776

Multi-Sense Language Modelling

Andrea Lekkas [1,2,3] , **Peter Schneider-Kamp** [1] , **Isabelle Augenstein**[2]
[1]Dept. of Mathematics and Computer Science, University of Southern Denmark
[2] Ordbogen A/S, Odense, Denmark
[3] Dept. of Computer Science, University of Copenhagen, Denmark
lekkas@imada.sdu.dk, petersk@imada.sdu.dk, augenstein@di.ku.dk

Abstract

The effectiveness of a language model is influenced by its token representations, which must encode contextual information and handle the same word form having a plurality of meanings (polysemy). Currently, none of the common language modelling architectures explicitly model polysemy. We propose a language model which not only predicts the next word, but also its sense in context. We argue that this higher prediction granularity may be useful for end tasks such as assistive writing, and allow for more a precise linking of language models with knowledge bases. We find that multi-sense language modelling requires architectures that go beyond standard language models, and here propose a structured prediction framework that decomposes the task into a word followed by a sense prediction task. For sense prediction, we utilise a Graph Attention Network, which encodes definitions and example uses of word senses. Overall, we find that multi-sense language modelling is a highly challenging task, and suggest that future work focus on the creation of more annotated training datasets.

1 Introduction

Any variant of language model, whether standard left-to-right, masked [Devlin *et al.*, 2019] or bidirectional [Peters *et al.*, 2018] has to address the problem of *polysemy*: the same word form having multiple meanings, as seen in Tab. 1. The meaning of a particular occurrence depends on the context, and all modern language modelling architectures from simple RNNs [Mikolov *et al.*, 2010] to Transformers use context-based representations. However, *token* representations in language models are not explicitly disambiguated. Single-prototype embeddings, i.e., traditional word vectors, have a 1-to-1 correspondence with word forms. Contextual embeddings change depending on the tokens in their context window, and are employed in recent models like ULMFit [Howard and Ruder, 2018], ELMo [Peters *et al.*, 2018], and all Transformer architectures. However, even for contextual embeddings, polysemy is handled in an implicit, non-discrete way: the sense that a word assumes in a particular occurrence is unspecified.

Sentence	Meaning
*"John sat on the **bank** of the river and watched the currents"*	**bank.n.01**: *sloping land, especially the slope beside a body of water*
*"Jane went to the **bank** to discuss the mortgage"*	**bank.n.02**: *a financial institution that accepts deposits and channels the money into lending activities*

Table 1: Example of polysemy. Senses taken from WordNet 3.0

Here, we propose the task of multi-sense language modelling, consisting of not word, but also sense prediction. We conjecture that multi-sense language models would:

1. improve the precision of linking a language model to a knowledge base, as done in [Logan *et al.*, 2019] to help generate factually correct language. For instance: "*The explorers descended in the cave and encountered a bat*" refers to the entity *'bat (animal)'* and not to *'bat (baseball implement)'*.
2. grant word prediction tools the ability to provide additional information, allowing them to show not only the predicted word but also its sense.
3. be useful in applications such as assistive writing, where it is desirable to display more information about a word sense to a user, such as its definition or usage.

Another potential use would be to explore if such dictionary information could improve standard language modelling, and reduce the number of training data needed, relevant for e.g. language modelling for low-resource languages. Consequently, our main research objectives are to:

- model next-sense prediction as task in addition to standard next-word prediction in language modelling, and examine the performance of different model architectures.
- encode sense background knowledge from sense definitions and examples, and examine how it can aid sense prediction.

As a sense inventory, we use WordNet 3.0 [Miller, 1995]. The sense background knowledge is encoded in a dictionary graph, as shown in Fig. 3. When reading a word w, the model can rely on an additional input signals: the state of the node that represents *w* in the dictionary graph (the *"global node"*). Node vectors in the graph are updated by a Graph Attention Network [Veličković *et al.*, 2018].

We find that sense prediction is a significantly more difficult task than standard word prediction. A way to tackle it is to use a structured prediction framework, where the next sense depends on the prediction of the next word. The most successful model we identified for this uses a hard cut-off for the number of words considered (SelectK, see § 4.2). The additional input signal from the dictionary graph provides only marginal improvements. For future work, we argue that larger sense-labelled datasets would go a long way towards improving the overall performance of multi-sense language models.

2 Related Work

We here review relevant works that disambiguate between word senses to address polysemy. They can be grouped in three categories: multi-prototype embeddings not connected to a knowledge base; supervised multi-sense embeddings based on a text corpus that only utilise a KB tangentially as the sense inventory; and models that rely more substantially on features from a KB, like glosses or semantic relations.

Multi-prototype embeddings The model of [Huang *et al.*, 2012] learns multi-prototype vectors by clustering word context representations. Single-prototype embeddings are determined by 2 FF-NNs with a margin objective on predicting the next word, a quasi-language modelling setting even if it utilises both the preceding and subsequent context. Multi-sense skip-gram [Neelakantan *et al.*, 2014] also defines senses as cluster centroids, measuring the cosine distance of the surrounding context of ±5 words. [Li and Jurafsky, 2015] use Chinese Restaurant Processes to decide whether to create a new cluster, and also investigate the usefulness of multi-sense embeddings in downstream tasks such as Semantic Relatedness and PoS tagging.

Supervised multi-sense embeddings Other models rely on sense-label supervision in a text corpus. *context2vec* by [Melamud *et al.*, 2016] builds contextual word embeddings by applying a biLSTM on text, and provides the option to create sense embeddings by using a sense-labelled corpus like Senseval-3 [Mihalcea *et al.*, 2004]. [Raganato *et al.*, 2017] is particularly relevant here: they frame WSD as a sequence learning problem, with the aim of finding sense labels for an input sequence. The training corpus is the same used in our work, SemCor [Miller *et al.*, 1993], and the core architecture is a biLSTM that reads both the preceding and subsequent context. A biLSTM is also employed in LSTMEmbed [Iacobacci and Navigli, 2019] to build word and sense representations; the sense-labelled training corpus is obtained by applying the BabelFly [Moro *et al.*, 2014] sense tagger to a text corpus that includes the English Wikipedia.

Most recently, SenseBERT [Levine *et al.*, 2020] uses WordNet supersenses (categories like food, artifact, person, etc.) to add a form of sense-prediction to BERT: an additional objective requires the model to predict one of the supersenses that the word *w* can assume. After training, the context information means the model can predict supersenses.

Sense representations can leverage glosses (definitions and examples) found in WordNet, as done in [Chen *et al.*, 2014] after single-prototype vectors are trained with a skip-gram model. Likewise, pre-trained word embeddings are the starting point for AutoExtend [Rothe and Schütze, 2015], an autoencoder architecture where word embeddings constitute the input and the objective whereas the embeddings for WordNet synsets are the intermediate encoded representation. [Kumar *et al.*, 2019] use a BiLSTM and self-attention to get contextual embeddings, then disambiguate them via a dot product with sense embeddings based on WordNet definitions.

KB-based methods In the last couple of years, efforts have been made to enable BERT to disambiguate between senses. GlossBERT [Huang *et al.*, 2019] takes in context-gloss pairs as input: for every target word in a context sentence, N=4 WordNet glosses are found; a classification layer determines which one of the possible lemmas the words can assume in the context sentence. SenseEmBERT [Scarlini *et al.*, 2020a] relies on the BabelNet mapping between WordNet and Wikipedia pages to collect relevant text for synsets, and computes the sense embeddings as a rank-weighted average of relevant synsets. ARES (Context Aware Embeddings of Senses [Scarlini *et al.*, 2020b]) goes through the same phases of context extraction, synset embeddings and sense embeddings: it computes lemma representations via BERT, and uses UKB [Agirre *et al.*, 2014] to create a set of contexts for the synset. Sense embeddings are obtained by concatenating the BERT representation of the sense contexts found in SemCor and the sense definition in WordNet.

Our model We use pre-trained embeddings and WordNet glosses and relations to create a dictionary graph. The vectors of sense nodes can be viewed as sense embeddings, although we are not interested in their quality since our primary objective is multi-sense language modelling. Future work may rely on them to improve sense disambiguation: for model variants that have to choose the correct sense among a limited number of candidates, there is an opening for the application of more complex multi-sense models than the ones explored here.

3 Multi-Sense Language Model

3.1 Architecture

A language modelling task decomposes the probability of predicting an entire text of length N as the product of each word prediction, where the probability of the next word $p(w_i)$ is influenced by the preceding context $[w_1, ..., w_{i-1}]$:

$$p(w_1, ..., w_N) = \prod_{i=1}^{N} p(w_i | w_1, ..., w_{i-1}) \tag{1}$$

Our model aims to carry out two language modelling tasks:

1. Standard language modelling: next-token prediction at the granularity of words.
2. Sense prediction: next-token prediction at the granularity of WordNet senses.

Therefore, the objective is to produce, at each position in the corpus, two probability distributions: one over the vocabulary of words and one over the vocabulary of senses.

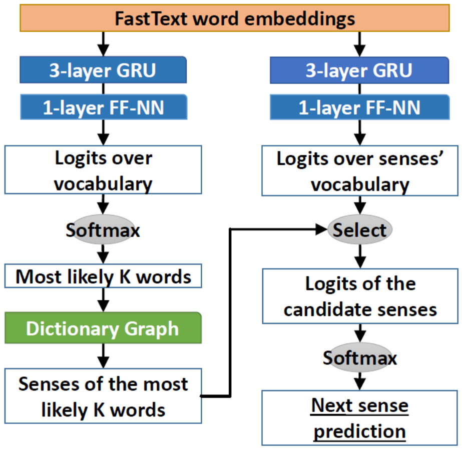

Figure 1: Overview of the SelectK version of the model. The input signal is in common for the word and the sense prediction task.

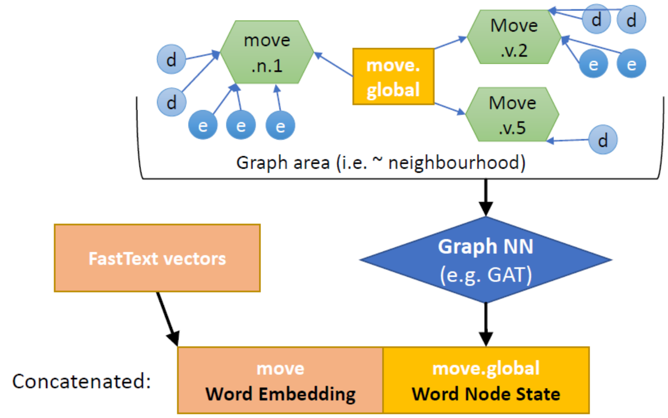

Figure 2: The input signals: the standard word embedding and the vector of the global node from the dictionary graph

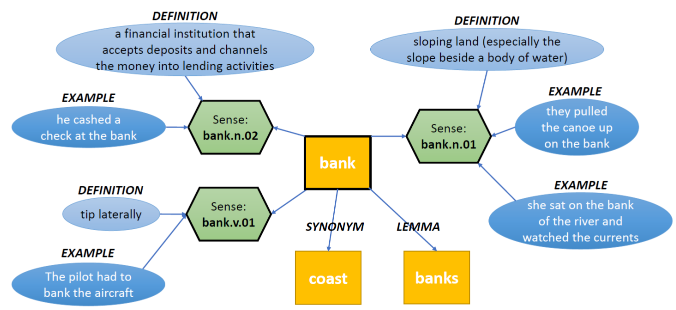

Figure 3: Part of the dictionary graph for the word *"bank"*. **Global** nodes are in yellow, sense nodes in green, definitions and examples in light blue and blue.

Reading each word w_t yields an input signal $I(w_t)$, that as specified in Fig. 2 consists of:

1. The single-prototype embedding for the word *w*, here, a pre-trained FastText vector [Bojanowski *et al.*, 2016]
2. The node-state of the *global node*, which represents *w* in the dictionary graph.

The function f performs the standard language modelling task, obtaining a probability distribution where $w^1_{t+1}, ..., w^K_{t+1}$ are the K words most likely to be the next token w_{t+1}; in all our architectures, f is implemented by a 3-layer GRU, followed by a FF-NN and softmax.
The sense prediction s_{t+1} is computed by a function g that always depends on the input, and in some model variants that use structured prediction (4.2, 4.4, 4.5) also depends on the K most likely words predicted by the standard language modelling task, $w^1_{t+1}, ..., w^K_{t+1}$. Fig. 1 showcases one of these structured prediction variants, SelectK (4.2).

$$\begin{aligned} w^1_{t+1}, ..., w^K_{t+1} &= f(I(w_t)) \\ s_{t+1} &= g(I(w_t), w^1_{t+1}, ..., w^K_{t+1}) \end{aligned} \tag{2}$$

The correctness of the prediction is evaluated using two measures: perplexity and accuracy (see Appendix A.3).

3.2 Dictionary Graph

First, we read in a text corpus and create the **vocabulary**. We register the lemmatised version of inflected forms, that are later connected to their parent forms. Then, we need to access the dictionary: for each sense of a word we retrieve the text of its **definitions** and **examples**, and register the connections with synonyms and antonyms. In the case of WordNet, senses are specified as the synsets of a word; example: w='bank' $\rightarrow$ [('bank.n.01'), ... ,('bank.v.07')] .

The next step is to compute the **sentence embeddings** for definitions and examples. It is possible to use any one of several methods, ranging from LSTMs to BERT's last layer. Given that the whole pipeline, comprising the retrieval of WordNet data, the creation of the dictionary graph and model training, takes a significant amount of time[1], for the sake of computation speed sentence embeddings are obtained as the average of the FastText word vectors.

Finally, the nodes are initialised and stored in a graph with their edges. The graph object is created with PyTorch-Geometric [Fey and Lenssen, 2019]; before training, it holds the initial values for the sense and global nodes. The **sense node** starts as the average of the embeddings of definitions and examples for that sense. As shown in Fig. 3, every sense node is directly connected to its definition and example nodes. Those determine its starting position; ideally, during training, it would be moved towards the more significant glosses. The **global node** is initialised as the FastText embedding for *w*.

3.3 Graph Attention Network

We employ a Graph Attention Network [Veličković *et al.*, 2018] to update the nodes of the dictionary graph, because unlike Graph Convolutional Networks [Kipf and Welling, 2017] based on the graph's Laplacian, the GAT is not bound to a specific graph structure, and unlike other methods like graphSAGE [Hamilton *et al.*, 2017] it can handle a variable number of neighbours. The underlying idea of GATs is to compute the representation vector h_i of node i based on its

[1] ranging from 8h14min to 31h27min. See Appendix ...

	Words		Senses	
	PPL	Accuracy	PPL	Accuracy
Vanilla GRUs	169.3	0.217	601.89	0.056
Vanilla GRUs w/ DG	164.7	0.219	585.47	0.104

Table 2: Baseline: two separate GRUs without any modification. Results of word prediction and sense prediction on SemCor's test set. **w/ DG**=including the graph node in the input signal

neighbouring nodes $j \in N(i)$, which have different attention weights (i.e. importance).

We here describe how a GAT obtains the new state h_i^{t+1} of node i in a graph with m nodes. The first step is to apply a linear transformation $\mathbf{W}$ over all the nodes: $\mathbf{W}h_1^t, ..., \mathbf{W}h_m^t$. Then, for each node j in the neighbourhood of i, we compute the non-normalised attention coefficient e_{ij}. The attention mechanism is a 1-layer FF-NN with LeakyReLU activation:

$$e_{ij} = LeakyReLU(\mathbf{A}^T[\mathbf{W}h_i, \mathbf{W}h_j]) \tag{3}$$

The normalised attention coefficients α_{ij} are obtained by applying a softmax over the neighbourhood of i, $N(i)$. Finally, the new state of node i is given by applying a non-linear function ρ to the weighted sum of the neighbours' states:

$$h_i^{t+1} = \rho\left(\sum_{j \in N(i)} \alpha_{ij}\mathbf{W}h_j^t\right) \tag{4}$$

[Veličković *et al.*, 2018] report that using multiple attention heads, by averaging or by concatenating, is beneficial to stabilise the model; we therefore use 2 concatenated heads.

3.4 Adjusting the Sense-Labelled Input

To train a multi-sense language model we need a sense-labelled corpus. Some of the words, like stopwords, will not have a sense label. If not handled, this would cause a problem because in some versions of the model (4.1, 4.2) the GRU used in the senses' task should be able to read the input text as an uninterrupted sequence of sense tokens. In particular, two types of words may have no sense specification:

1. **stopwords**: *'for'*, *'and'*, *'of'*, etc.
2. **inflected forms**: *'is'*, *'said'*, *'sports'*, etc.

In order to provide stopwords with a sense label, we add a **dummySense** (e.g. 'for.dummySense.01') for all the words without a sense. The corresponding graph node is initialised as the single-prototype FastText vector for that word. Inflected forms are **lemmatised** using NLTK's WordNetLemmatizer, to be able to read and predict the senses of their parent form ('is'→'be', 'said'→'say' etc.)

4 Architectures for Sense Prediction

As previously mentioned, multi-sense language modelling consists of the two tasks of standard language modelling and sense prediction, and we aim to output two probability distributions - for the next word token and the next sense token, respectively. With this aim, we devise several architectures.

4.1 Vanilla GRUs

Two ordinary GRU neural networks constitute our baseline. One network is dedicated to each task. There are no shared layers, the only element in common is the input signal: the pre-trained FastText word embedding for *w*, possibly concatenated with the state of *w*'s node from the dictionary graph as shown in Fig. 2. Among the model architectures for next-token prediction, we choose GRUs as they have a smaller number of parameters than either Transformers or LSTMs. Thus, they can be trained effectively on a text corpus as small as the one we used, the sense-labelled SemCor [Miller *et al.*, 1993] with 650K training tokens.

4.2 SelectK

SelectK is a structured prediction approach: choosing the next sense depends on the prediction of the next word. As the text is read, for every location t the standard language model outputs a probability distribution over the vocabulary, where the most likely K words are $w_1, ..., w_K$.

Every word w_i has a set of senses: $S(w_i) = \{s_{i1}, ..., s_{iN}\}$. The next sense at location t is chosen among the senses of the K most likely words at t:

$$s(t) \in \bigcup_{i=1}^{K} S(w_i) \tag{5}$$

Concretely, a softmax function is applied over the logits of the selected senses, while all other senses are assigned a probability $\epsilon = 10^{-8}$. The senses' logits are computed by the second GRU. K is a hyperparameter; K=1 means that the model always chooses among the senses of the most likely word. This means that the the sense prediction performance depends on the performance of the standard language model: if all the K most likely globals are incorrect, the correct sense cannot be retrieved. We verify what happens for K={1,5,10,50}.

4.3 Most Frequent Sense

This heuristic baseline chooses the *most frequent sense* found in the training set for the most likely word predicted by the standard language model.

4.4 Sense Context Similarity

This method, when making a prediction, pick as the candidates the senses of the most likely K globals, like SelectK. Then, they are ranked based on the cosine similarity between the local context and each sense's average context. Since language modelling is performed from left to right, the context is based only on the **preceding** tokens $[w_{t-1}, ..., w_{t-c}]$ without considering subsequent tokens.

For each occurrence $s_1, ...s_N$ of a sense s, the occurrence context $OC(s_i)$ is computed as the average of the word embeddings of the preceding *c* tokens. Afterwards, the Sense Context $SC(s)$ is computed as the average of the occurrences' contexts:

$$OC(s_i) = avg(w_{t-1}, ..., w_{t-c}) \tag{6}$$
$$SC(s) = avg(OC(s_1), ..., OC(s_N))$$

The representation of the local context can be either the average of the last *c* word embeddings, as shown here, or the output of a 3-layer GRU.

Method	Parameters K	C	context	All senses Correct	Accuracy	Senses of polysemous words Correct	Accuracy
SelectK	1	-	-	19036 / 88480	0.215	315 / 2535	0.012
MFS	1	-	-	18528	0.209	268	0.011
SenseContext	1	20	average	18006	0.204	225	0.009
Vanilla GRUs	-	-	-	4971	0.056	03	0
Self-Attention	1	10	average	16821	0,19	171	0.007
SenseContext	1	10	GRU	13009	0.147	14	0.001
SenseContext	5	20	average	12366	0.14	41	0.002
SelectK	5	-	-	10699	0.121	172	0.007
Self-Attention	5	10	average	7300	0.083	106	0.004

Table 3: The most relevant results of each method. Task: sense prediction on SemCor's test set. Sorted by the accuracy on all senses

4.5 Self-Attention Coefficients

Another way to choose among the candidate senses of the most likely K globals is to use the softmax scores from the self-attention mechanism. Every sense s has an average context $SC(s)$ it appears in, as seen in Eq. 6. The contexts of the candidate senses are collected in the matrix K. Then, a probability distribution over the senses is obtained by computing the self-attention coefficients:

$$softmax\left(\frac{Q \cdot K}{\sqrt{d_k}}\right) \tag{7}$$

All the rows of the query matrix Q are representations of the current context. As previously, the local context can be constructed as a simple average of the last c word embeddings, or as the output of a 3-layer GRU. The sense contexts in K take up the role of keys in the formula of self-attention scores.

5 Evaluation

5.1 Dataset and Experimental Settings

To train a multi-sense language model, we need a sense-labelled text corpus. We use **SemCor** [Miller *et al.*, 1993], a subset of the Brown Corpus labelled with senses from WordNet 3.0. Training, validation and test sets are obtained with a 80/10/10 split. We exclude all tokens with frequency=1 and we **lowercase** the text. The resulting dictionary graph has ≈120K nodes and ≈150K edges. Consequently, due to memory and time constraints, we apply mini-batching on the graph: for each input instance the Graph Attention Network operates on a local **graph area**. The graph area is defined expanding outwards from the global node of the current word w. In our experiments, a graph area contains at maximum 32 nodes and extends only for 1 hop, thus coinciding with the **neighbourhood** of the current word's global node.

To compute the logits, we employ two GRUs with 3 layers (1024→1024→512) followed by a Linear FF-NN (512→ |Vocabulary|). A grid-search is used to find reasonable **hyperparameters**: batch size={20, <u>32</u>, 40}; sequence length={<u>35</u>,70}; learning rate=$\{10^{-5}, \underline{5 \cdot 10^{-5}}, 10^{-4}\}$.

5.2 Model variants

In all tables (2, 3, 4) we list the following model variants:

- **Vanilla GRUs**: 2 separate GRUs, one for word prediction and one for sense prediction, sharing only the input signal. See Sec. 4.1.
- **SelectK**: Using 2 separate GRUs, but restricting the softmax over the senses'vocabulary to the *candidate senses* only: the senses of the most likely K words from the standard LM task. See Sec. 4.2.
- **SenseContext**: Using 1 GRU for word prediction, and choosing among the candidate senses based on the cosine similarity of local context and average sense context. See Sec. 4.4.
- **MFS**: Considering the most likely word predicted by the standard language model, choose its most frequent sense in the training set. See Sec. 4.3.
- **Self-Attention**: Using 1 GRU for word prediction, and choosing among the candidate senses by computing the self-attention coefficients of local context and average sense contexts. See Sec. 4.5.

The baseline model uses Vanilla GRUs (4.1): one GRU for word prediction and one for sense prediction, sharing only the input signal. The results are reported in Tab. 2. Given the differences in Perplexity (580 vs. 170) and accuracy (10% vs. 21%), predicting the next sense is significantly more difficult than standard language modelling. This highlights the need to find specific methods for sense prediction.

5.3 Results

In Tab. 3, we compare the results of the different methods for sense prediction. We report accuracy: **a)** on the senses of all words; **b)** only on the senses of polysemous words, i.e. those words that have more than 1 sense, which are an important part of any multi-sense task and are expected to be more difficult. We evaluate use accuracy instead of perplexity, since in structured prediction methods, perplexity values are non-significant ($\approx 10^8$) – this is caused by manually assigning $\epsilon = 10^{-8}$ as the probability of the non-candidate senses.

The best-performing method is SelectK with K=1, which is extremely reliant on the correctness of the word prediction task, and thus limited by the performance of the standard language model. We found that increasing K to 5 or 10 leads to a worse performance as seen in the first 2 rows of Tab. 3 and 4 and also in Appendix B, due to the increased difficulty of choosing among a greater number of candidate senses.

For predicting the right sense of a polysemous word, all methods have very low overall accuracy values, while the Vanilla GRU cannot perform the task at all, resulting in an accuracy of 0. We surmise that improvements can be achieved

Method	Parameters K	C	context	All senses Accuracy	DG Δ	Senses of polysemous words Accuracy	DG Δ
SelectK	1	-	-	0.217	+0.002	0.013	+0.001
SelectK	5	-	-	0.128	+0.009	0.006	-0.001
SenseContext	1	20	average	0.205	+0.001	0.011	+0.002
SenseContext	5	20	average	0.086	-0.054	+0.006	+0.004
SenseContext	1	10	GRU	0.135	-0.012	0	-0.001
Self-Attention	1	10	average	0.192	+0.002	0.009	+0.002
Self-Attention	5	10	average	0.088	+0.005	0.003	-0.001
Vanilla GRUs	-	-	-	0.104	0	0	0
MFS	-	-	-	18637	0.211	0.012	+0.001

Table 4: Sense prediction on SemCor's test set. **DG Δ** = change in accuracy caused by including the input from the dictionary graph.

in the structured prediction framework, either by training the model differently – as mentioned later, by better disambiguating between the candidate senses, or, indeed, by utilising a larger training dataset, which currently does not exist.

Inclusion of Dictionary Graph Input Up to now, the word and sense prediction tasks share the same input signal. This set of experiments test if concatenating the state of a word *w*'s node computed by the GAT with the FastText word embedding results in an increase in performance.

As can be seen in Tab. 4, the graph input signal results in an extremely small increase in performance for the SelectK and Self-Attention methods, while it is has no impact on Vanilla GRUs and is silghtly detrimental to SenseContext. The ablation studies in Appendix B confirm this behaviour across the different methods and hyperparameters. Future work may find improvements by examining the use of different Graph Neural Network architectures, or different ways of encoding dictionary definitions and examples in the dictionary graph.

6 Discussion

Next-token prediction at the granularity of senses is a more difficult task than standard language modelling, due to operating on a larger vocabulary with a more extensive low-frequency long tail. Moreover, there are relatively few sense-labelled datasets. The datasets organised in UFSAC format [Vial *et al.*, 2018] altogether contain 44.6M words, of which only 2.0M are annotated; in SemCor 29.4% of the tokens have a sense label. These datasets are available for English only, thus studying the benefit of integrating sense-labelled corpora and dictionary resources for low-resource languages cannot currently be pursued until such corpora are created.

Overall, using a structured prediction framework for multi-sense language modelling is more promising than applying a GRU directly to next-sense prediction, allowing one to choose a sense among a small number of candidate senses.

However, none of the methods studied here achieve an accuracy greater than 22%, and the best results are obtained by choosing among the senses of the most likely word. If a sense prediction method managed to reliably choose among a higher number of candidate senses, it would make the sense prediction task less dependant on achieving a good performance for the standard language modelling task. The question of what such a method could be remains open. It may be solved either by investigating other methods, such as different ways of encoding the dictionary graph, or by changing the training procedures for the methods proposed here – e.g., one could freeze the part of the architecture for standard language modeling and then separately optimise only the sense prediction part. Moreover, one could expect that next-token prediction, both at word and sense level, would benefit from using architectures more advanced than a GRU, such as Transformer architectures. These are, however, more parameter-expensive and would require training on a larger sense-labelled corpus than SemCor. This consideration is validated by the fact the LSTM variants are the SoA architecture on a small standard language modelling dataset such as WikiText-2 [Merity *et al.*, 2016], instead of Transformers [Melis *et al.*, 2020].

Including the input signal from the dictionary graph only results in marginal improvements on the sense prediction task. It should be noted that the quality of the input signal is limited by the quality of the sentence encodings for the WordNet glosses, used to initialise the graph nodes. Sentence representations different from averaging FastText embeddings may achieve better results. Moreover, tuning the graph signal is surely possible, while outside of the scope of this first study of multi-sense language modelling: one could experiment with changing the size of the graph area, the number of hops and the variant of Graph Neural Network used.

7 Conclusions

This work constitutes the first study of multi-sense language modelling, and highlights its difficulty. We experiment with a structured prediction approach that predicts a word followed by a word sense; as well as learning sense representations from both its sentential context, and a sense dictionary, encoding the latter using a Graph Neural Network. Future work on this task may view it as a test bed for researching Word Sense Disambiguation, as a way of improving the precision of linking a language model to a knowledge base, or for applications such as assistive writing. Finally, we note that significant progress in terms of absolute performance is currently hampered by the low availability of sense-labelled resources, and suggest coordinated efforts on this for future work.

References

[Agirre *et al.*, 2014] Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84, March 2014.

[Bojanowski *et al.*, 2016] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[Chen *et al.*, 2014] Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035, Doha, Qatar, October 2014. Association for Computational Linguistics.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[Fey and Lenssen, 2019] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[Hamilton *et al.*, 2017] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017.

[Howard and Ruder, 2018] Jeremy Howard and Sebastian Ruder. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146, 2018.

[Huang *et al.*, 2012] Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[Huang *et al.*, 2019] Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. GlossBERT: BERT for word sense disambiguation with gloss knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3509–3514, Hong Kong, China, November 2019. Association for Computational Linguistics.

[Iacobacci and Navigli, 2019] Ignacio Iacobacci and Roberto Navigli. LSTMEmbed: Learning word and sense representations from a large semantically annotated corpus with long short-term memories. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1685–1695, Florence, Italy, July 2019. Association for Computational Linguistics.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

[Kumar *et al.*, 2019] Sawan Kumar, Sharmistha Jat, Karan Saxena, and Partha Talukdar. Zero-shot word sense disambiguation using sense definition embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5670–5681, Florence, Italy, July 2019. Association for Computational Linguistics.

[Levine *et al.*, 2020] Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. SenseBERT: Driving some sense into BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667, Online, July 2020. Association for Computational Linguistics.

[Li and Jurafsky, 2015] Jiwei Li and Dan Jurafsky. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[Logan *et al.*, 2019] Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5962–5971, Florence, Italy, July 2019. Association for Computational Linguistics.

[Melamud *et al.*, 2016] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany, August 2016. Association for Computational Linguistics.

[Melis *et al.*, 2020] Gábor Melis, Tomás Kociský, and Phil Blunsom. Mogrifier LSTM. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[Merity *et al.*, 2016] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *CoRR*, abs/1609.07843, 2016.

[Mihalcea *et al.*, 2004] R. Mihalcea, T. Chklovski, and A. Kilgarriff. The Senseval-3 English lexical sample task. In *Proceedings of SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text [CD-ROM]*, pages 25–28, 2004.

[Mikolov *et al.*, 2010] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura, editors, *INTERSPEECH*, pages 1045–1048. ISCA, 2010.

[Miller *et al.*, 1993] George A. Miller, Claudia Leacock, Ee Tengi, and Ross T. Bunker. A semantic concordance. In *Proceedings ARPA Human Language Technology Workshop*, pages 303–308, 1993.

[Miller, 1995] George A. Miller. Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41, 1995.

[Moro *et al.*, 2014] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014.

[Neelakantan *et al.*, 2014] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar, October 2014. Association for Computational Linguistics.

[Peters *et al.*, 2018] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[Raganato *et al.*, 2017] Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[Rothe and Schütze, 2015] Sascha Rothe and Hinrich Schütze. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China, July 2015. Association for Computational Linguistics.

[Scarlini *et al.*, 2020a] Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. Sensembert: Context-enhanced sense embeddings for multilingual word sense disambiguation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8758–8765, Apr. 2020.

[Scarlini *et al.*, 2020b] Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. With More Contexts Comes Better Performance: Contextualized Sense Embeddings for All-Round Word Sense Disambiguation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2020.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[Vial *et al.*, 2018] Loïc Vial, Benjamin Lecouteux, and Didier Schwab. UFSAC: Unification of Sense Annotated Corpora and Tools. In *Language Resources and Evaluation Conference (LREC)*, Miyazaki, Japan, May 2018.

A Further information

A.1 SemCor corpus

Training tokens: 646,038
Validation tokens: 80,760
Test tokens: 78,453
Number of words in the vocabulary=24,689

A.2 Graph & Glosses

Graph
Sense nodes = 40095
Global (word) nodes = 24689
Dictionary definition nodes = 33568
Dictionary example nodes = 29585
Edges = 217329

Glosses
For all the words *w* in the SemCor corpus, definitions and examples were retrieved from the synsets in WordNet 3.0 where *w* is the lemma of the first word.
The length of glosses is computed by eliminating punctuation and splitting on whitespace.

Average # of tokens in a definition = 8.83; max.= 69.
Average # of tokens in an example = 5.95; max.= 33.

A.3 Evaluation measures

Perplexity
Given a test set $T = (w_1, w_2, w_3, ...w_N)$, and a language model L that assigns a probability $P(w_i)$ to each word w_i in a given position, the *perplexity* of L on T is defined as *the inverse probability of encountering the words in a test set, normalised by the number of words in the set*:

$$PP(T) = \frac{1}{P(w_1 w_2 ... w_n)^{1/N}} = \sqrt[\frac{1}{N}]{\frac{1}{P(w_1 w_2 ... w_n)}} \quad (8)$$

This measure coincides with elevating 2 to the Shannon entropy value $H(p)$ for the probability distribution:

$$PP(T) = 2^{H(p)} = 2^{-\sum_i p(w_i) \cdot \log_2(p(w_i))} \quad (9)$$

Accuracy
The number of correct predictions divided by the total number of tokens to predict, i.e. $\frac{correct}{total}$

A.4 Hardware

Each experiment was executed on 1 NVidia GPU:
GeForce RTX 2080 Ti, with 10.76 GB of memory.
CPU(s): 20 x IntelCore i9-9820X CPU @ 3.30GHz.
RAM: 125 GB

CUDA Version: 10.1
PyTorch version: 1.5.0+cu101

B Further experiments

Tables follow on the next page.

Input	Parameters	Senses (all words)		Senses (only polysemous words)	
	K	N.correct	Accuracy	N.correct	Accuracy
FastText embeddings only	1	19036 / 88480	0,215	315 / 25353	0,012
	5	10699	0,121	172	0,007
	10	7146	0,081	71	0,003
	50	3095	0,035	37	0,001
Concatenated: FT embeddings + word node state	1	19178 / 88480	0,217	340 / 25353	0,013
	5	11311	0,128	145	0,006
	10	9678	0,109	74	0,003
	50	2804	0,032	48	0,002

Table 5: Results of the **SelectK** method. Task: sense prediction on SemCor's test set.

Input	Method	Senses (all words)		Senses (polysemous words)	
		N.correct	Accuracy	N.correct	Accuracy
FastText embeddings only	Vanilla GRU	9192 / 88480	0.104	0 / 25353	0
	MFS	18528	0.209	268	0.011
Concatenated: FT embeddings + word node state	Vanilla GRU	9192	0.104	0	0
	MFS	18637	0.211	293	0.012

Table 6: Results of the non-parametric methods **VanillaGRU** and **Most Frequent Sense**. Task: sense prediction on SemCor's test set

Input	Parameters			Senses (all words)		Senses (polysemous words)	
	context	K	C	N.correct	Accuracy	N.correct	Accuracy
FastText embeddings only	Average	1	10	16821 / 88480	0.19	171 / 25353	0.007
	Average	1	20	16417	0.186	163	0.006
	Average	5	10	7300	0.083	106	0.004
	Average	5	20	7042	0.08	29	0.001
Concatenated: FT embeddings and node state	Average	1	10	16989	0.192	237	0.009
	Average	1	20	16563	0.187	165	0.007
	Average	5	10	7769	0.088	86	0.003
	Average	5	20	7435	0.084	40	0.002

Table 7: Results of the **Self Attention Scores** method. Task: sense prediction on SemCor's test set

Input	Parameters			Senses (all words)		Senses (polysemous words)	
	context	K	C	N.correct	Accuracy	N.correct	Accuracy
FastText embeddings only	Average	1	10	17300 / 88480	0.196	62 / 25353	0.002
	Average	1	20	18006	0.204	225	0.009
	Average	5	10	11039	0.125	34	0.001
	Average	5	20	12366	0.14	41	0.002
	GRU	1	10	13009	0.147	14	0.001
	GRU	1	20	12625	0.143	16	0.001
	GRU	5	10	4058	0.046	15	0.001
Concatenated: FastText embeddings + node state	Average	1	10	17234 / 88480	0.195	47 / 25353	0,002
	Average	1	20	18106	0.205	282	0,011
	Average	5	10	5967	0.067	52	0,002
	Average	5	20	7653	0.086	161	0,006
	GRU	1	10	11940	0.135	12	0
	GRU	1	20	12061	0.136	15	0.001
	GRU	5	10				

Table 8: Results of the **SenseContext** method. Task: sense prediction on Semcor's test set.