Neural Networks 142 (2021) 397–409

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Unsupervised multi-sense language models for natural language processing tasks

Jihyeon Roh [a], Sungjin Park [a], Bo-Kyeong Kim [b], Sang-Hoon Oh [c], Soo-Young Lee [a,*]

[a] School of Electrical Engineering and Institute for Artificial Intelligence, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea
[b] Information & Electronics Research Institute, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea
[c] Division of Information and Communication Convergence Engineering, Mokwon University, Daejeon, Republic of Korea

ARTICLE INFO

Article history:
Received 22 August 2020
Received in revised form 27 April 2021
Accepted 19 May 2021
Available online 25 May 2021

Keywords:
Language model
Neural language processing (NLP)
Multi-sense word modeling

ABSTRACT

Existing language models (LMs) represent each word with only a single representation, which is unsuitable for processing words with multiple meanings. This issue has often been compounded by the lack of availability of large-scale data annotated with word meanings. In this paper, we propose a sense-aware framework that can process multi-sense word information without relying on annotated data. In contrast to the existing multi-sense representation models, which handle information in a restricted context, our framework provides context representations encoded without ignoring word order information or long-term dependency. The proposed framework consists of a context representation stage to encode the variable-size context, a sense-labeling stage that involves unsupervised clustering to infer a probable sense for a word in each context, and a multi-sense LM (MSLM) learning stage to learn the multi-sense representations. Particularly for the evaluation of MSLMs with different vocabulary sizes, we propose a new metric, i.e., unigram-normalized perplexity (*PPLu*), which is also understood as the negated mutual information between a word and its context information. Additionally, there is a theoretical verification of *PPLu* on the change of vocabulary size. Also, we adopt a method of estimating the number of senses, which does not require further hyperparameter search for an LM performance. For the LMs in our framework, both unidirectional and bidirectional architectures based on long short-term memory (LSTM) and Transformers are adopted. We conduct comprehensive experiments on three language modeling datasets to perform quantitative and qualitative comparisons of various LMs. Our MSLM outperforms single-sense LMs (SSLMs) with the same network architecture and parameters. It also shows better performance on several downstream natural language processing tasks in the General Language Understanding Evaluation (GLUE) and SuperGLUE benchmarks.

1. Introduction

Language models (LMs) have been core elements in numerous applications of natural language processing (NLP), e.g., language modeling (Bengio, Ducharme, Vincent, & Jauvin, 2003; Mikolov, Kombrink, Burget, Černockỳ, & Khudanpur, 2011), machine translation (Cho et al., 2014), and speech recognition (Amodei et al., 2016). LMs determine the probability of word sequences and are designed to generate high-probability sequences that are both semantically and synthetically meaningful. To achieve good performance, LMs must accurately capture the relationships among words and phrases in word sequences.

Compared with classical count-based *n*-gram LMs (Kneser & Ney, 1995), recent neural-network-based LMs (NLMs) (Bengio et al., 2003; Jozefowicz, Vinyals, Schuster, Shazeer, & Wu, 2016; Kim, Jernite, Sontag, & Rush, 2016) have performed better to encode the relationships among words based on vector representations, also known as word embeddings. In particular, recurrent neural networks (RNNs) (Elman, 1990), which can effectively process long-term time dependency, have been spotlighted. Despite their superior performance in several applications, most NLMs simply learn a single embedding vector per word while neglecting words that have different meanings depending on the context. For example, multiple meanings of "apple" (i.e., the fruit or the company) or "bank" (i.e., a financial institution or sloping land) are contained in one embedding and cannot be distinguished.

To address the ambiguity of word embeddings, previous studies have distinguished the meanings of multi-sense words based

* Corresponding author.
E-mail addresses: rohleejh@kaist.ac.kr (J. Roh), zxznm@kaist.ac.kr (S. Park), bokyeong1015@gmail.com (B.-K. Kim), shoh@kaist.ac.kr (S.-H. Oh), sylee@kaist.ac.kr (S.-Y. Lee).

https://doi.org/10.1016/j.neunet.2021.05.023
0893-6080/

on their surrounding context. In many previous studies (Bartunov, Kondrashkin, Osokin, & Vetrov, 2016; Li & Jurafsky, 2015; Neelakantan, Shankar, Passos, & McCallum, 2014; Qiu, Tu, & Yu, 2016), a word with multiple senses has been explicitly embedded into several distinct vectors (i.e., explicit sense vectors) for distributed representation models, such as continuous bag-of-words (CBoW) (Mikolov, Chen, Corrado, & Dean, 2013) and skip-gram (SG) (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) models. There are some studies (Aharoni & Goldberg, 2020; Ansell, Bravo-Marquez, & Pfahringer, 2021; Ma, Jin, Liu, Chen, & Yu, 2020) that model explicit sense vectors using neural LMs. However, these previously works suffer from several drawbacks. The composition of the context from distributed representation models (i.e., word2vec models) readily captures word relationships but has limited the number of neighboring words that comprise the context (Bartunov et al., 2016; Li & Jurafsky, 2015; Neelakantan et al., 2014; Qiu et al., 2016). In addition, due to the inclusion of future words and neglecting of the order of words in the context, it is difficult to directly apply the learned context representations to language modeling tasks. Although predefined topics help simplify the learning algorithm, it may produce some meaningless (unnatural) representations (Aharoni & Goldberg, 2020; Neelakantan et al., 2014). Also, the mixing of word- and sense-level tokens enables the optimization of the entire learning process with a single model, but word-level tokens lead to ambiguity in the representation learning process (Ansell et al., 2021; Li & Jurafsky, 2015; Ma et al., 2020; Neelakantan et al., 2014; Zhao & Mao, 2017). Therefore, the performance improvement of the multi-sense embedding was limited to simpler LMs and not eminent for advanced LM models such as LSTM (Li & Jurafsky, 2015).

Recently, contextualized word embeddings (CWEs) have been introduced to implicitly learn word representations based on the context (Devlin, Chang, Lee, & Toutanova, 2019; Peters et al., 2018; Radford, Narasimhan, Salimans, & Sutskever, 2018). These methods are built on the basis of advanced neural networks, such as LSTM networks (Hochreiter & Schmidhuber, 1997) and Transformers (Vaswani et al., 2017), and have achieved remarkable performance on several NLP tasks. However, they still use one embedding per word as the input, which leaves a very difficult task of distinguishing multiple senses to the neural network models.

In this paper, by combining the explicit unsupervised clustering of multi-sense words with advanced NLMs, we propose a flexible framework to achieve improved LM performance. Here, we focus on two specific types of advanced NLMs, i.e., LSTM and Transformer based networks, which have been actively explored in the NLP field recently. Specifically, our framework consists of 3 stages, i.e., (a) a *context representation learning stage* to encode the variable-size context into a dense representation, (b) a *sense-labeling stage by unsupervised clustering* to infer a probable sense based on the learnt representation, and (c) a *multi-sense LM (MSLM) learning stage* to learn multi-sense representations with the inferred senses. Words with different inferred senses are treated as different tokens and fed to either unidirectional or bidirectional LMs. Both LSTM and Transformer networks are implemented in the LMs.

We further estimate the number of clusters (senses) to improve the robustness of the learning algorithm. To compare LM performance given different vocabulary sizes, we present a new evaluation metric, i.e., unigram-normalized perplexity (*PPLu*). We demonstrate quantitative improvements over the single-sense LM (SSLM) and recent CWE methods on the SemCor all-words (SemCor), Penn Treebank (PTB), and Text8 datasets and several downstream NLP tasks. Qualitative evaluations of the nearest-neighbor words show the superiority of our MSLMs. Our contributions are summarized as follows:

- We propose a novel framework for the effective integration of purely unsupervised word sense clustering with sequence-based unidirectional or bidirectional NLMs.
- We propose a new LM performance metric that is not affected by vocabulary size.
- We quantitatively and qualitatively demonstrate the superiority of our multi-sense method over SSLMs on several natural language understanding (NLU) benchmark tasks.

This paper is organized as follows. After reviewing the related works in Section 2, we introduce the proposed three-stage learning framework in Section 3. The datasets and experimental settings used are described in Section 4. Experimental results on several datasets and tasks are presented and analyzed in Section 5. Finally, our concluding remarks are summarized in Section 6.

2. Related works

Since our study is based on LMs and multi-sense word modeling, some relevant previous studies are reviewed here.

2.1. Statistical language models

The aim of statistical LMs is to compute the probability distribution of sequences in various language units (e.g., characters or words). These LMs can be classified into two categories, i.e., count-based LMs and NLMs. Traditional count-based methods generally estimate n-gram probability based on Markov assumptions using counting and smoothing techniques (Kneser & Ney, 1995). These LMs rely on patterns of symbolic token sequences, in which the similarities among words are not considered. NLMs solve this problem by using the distributed representation of language units (e.g., word-embedding vectors).

NLMs have two main streams, i.e., finite-size n-gram NLMs and sequence-based LMs. The n-gram NLMs were proposed to solve the above problems of symbolic token counts (Bengio et al., 2003; Cheng, Kok, Pham, Chieu, & Chai, 2014; Pham, Kruszewski, & Boleda, 2016; Zhang, Jiang, Xu, Hou, & Dai, 2015). Additionally, sequence-based LMs have increased the context size to almost infinity (Jozefowicz et al., 2016; Mikolov et al., 2011; Vaswani et al., 2017). Unlike traditional methods, network parameters, including word embeddings, are learned jointly in the learning process.

The first approach for an n-gram NLM was to develop a neural probabilistic LM (Bengio et al., 2003) built on a simple feed-forward neural network (FNN). Based on $(n-1)$ previous words an n-gram model produced a conditional probability distribution of the next word. Then, a convolutional neural network (CNN)-based LM (Pham et al., 2016) was proposed as an advanced model. However, they take a fixed number of previous words as input, and therefore n-gram NLMs face difficulty in processing distant context information.

Conversely, sequence-based LMs were built with RNNs and/or Transformer models to transfer information for a long time. For RNNs each hidden state is represented by a composition of all previous words, and the final hidden state may represent the whole sentence. Recently, advanced forms of RNNs, such as LSTM, have been used to reduce the vanishing gradient problem. Some studies have proposed the application of several learning techniques (e.g., variational dropout, weight tying, and augmented loss function (Inan, Khosravi, & Socher, 2016)) to LSTM and added caches that enhance information storage capacity (Grave, Joulin, & Usunier, 2016).

Unlike previous works that utilize only word-level tokens, some studies have incorporated subword information to represent words (Botha & Blunsom, 2014; Kim et al., 2016). For example, a character-aware NLM (Kim et al., 2016) builds the structural

characteristics of the words with CNNs, which take character-level tokens as the input and generate outputs to LSTM-based LMs.

An attention-based sequential modeling architecture, Transformer (Vaswani et al., 2017), has gained attention due to its parallel computation and interpretability. As representative Transformer-based models, bidirectional encoder representations from transformers (BERT) (Devlin et al., 2019) and generative pretraining (GPT) (Radford et al., 2018, 2019) models pretrained multi-layer Transformers on a large corpus and demonstrated superior performance on many language processing tasks for much longer sequences. A variety of BERT and GPT variant models, like XLNet (Yang et al., 2019), have emerged. Our work focuses on LSTM-based and Transformer-based LMs to address long-term context information. However, the embedding ambiguity of multi-sense words still remains. To alleviate this limitation, we combine the multi-sense word embedding method with sequence-based NLMs.

2.2. Modeling multi-sense words

Several approaches have been proposed for modeling multi-sense words (Bartunov et al., 2016; Guo, Che, Wang, & Liu, 2014; Li & Jurafsky, 2015; Neelakantan et al., 2014; Qiu et al., 2016; Reisinger & Mooney, 2010; Song, Wang, Mi, & Gildea, 2016). In sense representation, each sense of a word needs to be represented differently depending on its context. Related studies can be classified into three categories, i.e., supervised learning methods with a sense-annotated corpus, knowledge-based disambiguation methods, and unsupervised learning methods.

Studies using supervised learning methods have focused on learning the word sense disambiguation (WSD) from examples. These works (Melamud, Goldberger, & Dagan, 2016; Yuan, Richardson, Doherty, Evans, & Altendorf, 2016) map words to their related senses using a sense-annotated corpus. Knowledge-based research has begun to achieve good progress in the development of large lexical resources such as WordNet (Miller, Beckwith, Fellbaum, Gross, & Miller, 1990) and BabelNet (Navigli & Ponzetto, 2012). For example, translation-based methods (Guo et al., 2014), which rely on a bilingual parallel database, have shown good performance in word sense induction. However, difficulties have arisen in applying knowledge-based methods to languages with insufficient lexical resources. Both supervised learning methods and translation-based methods have encountered difficulties in obtaining the training data. Due to these difficulties, our work focuses on an unsupervised learning method that requires only a monolingual database without multi-sense annotation.

Unsupervised learning methods aim to assign proper multiple senses for a multi-sense word from unannotated text data. These works infer multiple senses of words, including the types of words (e.g., nouns and verbs) used in the WSD task. Even if the meaning of a word has changed over time, these methods can cope with the change. They usually consist of two parts. The first part involves defining a context for a word and computing its context representation. The second part involves grouping context representations according to their similarity and relabeling a word token with one of several corresponding senses.

While the studies reported in Ansell et al. (2021), Ma et al. (2020), Neelakantan et al. (2014), Reisinger and Mooney (2010) considered a fixed number of sense representations for all words, there were a few studies that used different numbers of sense representations for each word. The nonparametric multi-sense skip-gram (MSSG) model (Neelakantan et al., 2014) and skip-gram (SG) with the Chinese restaurant process (CRP) (Li & Jurafsky, 2015) determine the number of senses for each word. Zhao and Mao (2017), Jain, Bodapati, Nallapati, and Anandkumar (2019), and Panigrahi et al. (2019) assigned one of the predefined topics to each word to distinguish the meanings of the multi-sense words, where the topic is derived from topic modeling methods such as latent dirichlet allocation (LDA) (Blei, Ng, & Jordan, 2003). Panigrahi et al. (2019) utilized a generative model and topic distribution to learn an interpretable word sense representation. Aharoni and Goldberg (2020) proved the existence of clusters in pretrained LMs that capture domain-specific information contained in a sentence. These results justified the replacement of the sense with a topic. PolyLM (Ansell et al., 2021) added the disambiguation encoder to the vanilla Transformer encoder. The disambiguation encoder was designed to represent the subword as the weighted sum of the predefined senses and to allocate the correct sense to each subword.

Our proposed framework differs from the abovementioned works. First, previous works (Bartunov et al., 2016; Neelakantan et al., 2014; Qiu et al., 2016; Song et al., 2016) computed context representations by averaging word embeddings within a local context of a fixed window size while ignoring any words outside this window. In our framework, the applied sequence-based LMs utilize the full context length with word order information. Second, one previous study (Guo et al., 2014) considered contexts only from previous words, while we integrate multi-sense words into LMs with both unidirectional and bidirectional LSTMs and Transformers. Third, our framework does not require any topic modeling and predefined topics used in previous works (Aharoni & Goldberg, 2020; Jain et al., 2019; Panigrahi et al., 2019).

Another group of methods based on CWEs (Dai et al., 2019; Devlin et al., 2019; Peters et al., 2018; Radford et al., 2018; Yang et al., 2019) focuses on representing the meaning of a word by using its context and does not require an explicit sense induction step. The methods in these studies relied on the implicit modeling of the senses of words using LMs and outperformed previous methods for recent downstream NLP tasks.

In the embeddings from LMs (ELMo) (Peters et al., 2018), CWEs are obtained from a bidirectional LM based on character-level LSTMs. A linear combination of representations from each hidden layer is used as the context representations. Unidirectional or bidirectional Transformer-based models such as GPT and BERT represent CWEs by encoding context information through an attention analysis of neighboring words. There have been a number of other methods based on CWE for the word sense disambiguation task (Huang, Sun, Qiu, & Huang, 2019; Vial, Lecouteux, & Schwab, 2019). However, these works do not produce explicit sense representations.

The aforementioned works share the same philosophy with our work. Both works aim to encode word senses from context information in LM. However, we explicitly separate the multi-sense disambiguation stage from the LM stage. By making each stage separate, we are able to easily adopt any state-of-the-art LMs such as GPT, BERT, and XLNet. In addition, for downstream NLP tasks, it is possible to import only the pretrained sense embeddings into task-specific networks, whereas all hidden-layer features of the pretrained LM are readily available without many additional computational costs.

3. Method

Fig. 1 depicts our proposed framework, which consists of three main stages, i.e., *context representation via single-sense LM learning* in Stage 1, *sense labeling via unsupervised clustering* in Stage 2, and *multi-sense LM learning* in Stage 3. Stage 1 encodes the context information of a word into a single representation. Stage 2 determines the sense of a word among its possible senses given its context representations. The context representations

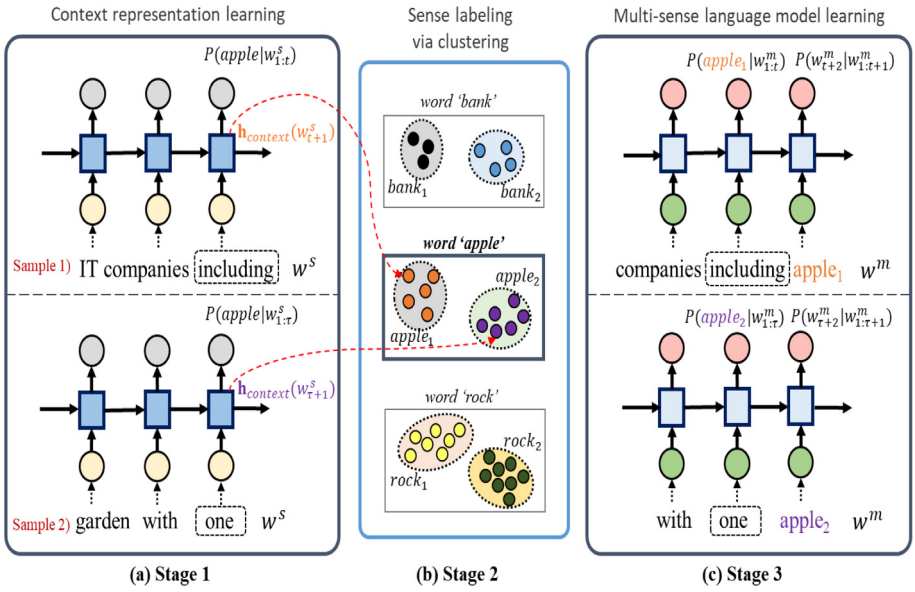

Fig. 1. The proposed framework consists of three stages. (a) In Stage 1, a single-sense LM learns context representation vectors ($\mathbf{h}_{context}(w^s)$) from word tokens w^s (i.e., a single sense per word). (b) In Stage 2, the learned context vectors of each multi-sense word in a training corpus are split up by an unsupervised clustering algorithm to create several new word-sense tokens w^m, e.g., "apple$_1$" and "apple$_2$". For simplicity, the number of clusters, K, is shown to be 2. (c) In Stage 3, a multi-sense LM learns context representation vectors from the word-and-sense tokens, w^m. Although these clustering in (b) and multi-sense LM learning in (c) may be repeated for iterative refinement, results from only one learning cycle are reported in this paper.

from Stage 1 are grouped depending on their similarity in Stage 2, where one word is assumed to have an arbitrary K number of senses (i.e., word meanings). Stage 3 encodes the clustered inferred word senses via another LM learning. Various neural LMs can be utilized in Stages 1 and 3.

Stage 1 uses a single representation per word that is associated with word-level tokens w^s, while Stage 3 uses a multi-sense representation that is associated with sense-level tokens w^m, e.g., apple$_1$ or apple$_2$ in Fig. 1(c) for the word "apple". In Stages 1 and 3, either a unidirectional or bidirectional LM can be used based on either LSTMs or Transformers. In the following section, we denote vocabulary in uppercase italics V^s and V^m for the single-sense and multi-sense representations, respectively. Each word sense is assumed to have one index within the vocabulary. The whole procedure is summarized in **Algorithm 1**.

Algorithm 1: Multi-Sense LM Algorithm

```
# Stage 1
Optimize a single-sense LM with a vocabulary (v^s_1, ..., v^s_|V^s|)
# Stage 2
for v^s in vocabulary V^s do
    # Compute context vectors of v for all R_v occurrences in the corpus
    H_v = {h_context(v^s(1)), ..., h_context(v^s(R_v))}
    Compute μ^v_1, ..., μ^v_{K_v} using k-means clustering on H_v
    Assign a cluster as z_v(r) = arg min_{k=1,...,K_v} ||h_context(v^s(r)) − μ^v_k||
end for
# Stage 3
Initialize multi-sense vocabulary (v^m_1, ..., v^m_|V^m|) by expanding all z_v(r) into one-dimensional array
Replace all single-sense tokens w^s_{1:T} with multi-sense tokens w^m_{1:T} in the corpus
Optimize a multi-sense LM with a new vocabulary (v^m_1, ..., v^m_|V^m|)
```

3.1. Stage 1: Context representation via single-sense LM learning

Fig. 1(a) depicts the unidirectional LM for learning context vector representations. A context vector is a representation that encodes the meaning of a word. For a word w_t at time t in a sentence $w_{1:T} = \{w_1, \ldots, w_T\}$, its context vector often summarizes all other words, which include past words $\{w_1, \ldots, w_{t-1}\}$ and future words $\{w_{t+1}, \ldots, w_T\}$. Vocabulary size $|V^s|$ is the number of unique words that appear in the training corpus. Depending on how to define the context, Stage 1 can be built upon either a unidirectional LM using only past words or a bidirectional LM using both past and future words.

When token w_t^s is a word, v, in V^s at time t, the input to the L-layer LM is an N-dimensional single-sense word representation, $\mathbf{x}_t$, which is one column of the single-sense embedding matrix $\mathbf{X} \in \mathbb{R}^{N \times |V^s|}$. At each time step t at the l-th hidden layer, hidden vector $\mathbf{h}_t^l$ is calculated from lower-layer hidden vector $\mathbf{h}_t^{l-1}$ and the previous hidden state vector, $\mathbf{h}_{t-1}^l$. For the lowest layer, $\mathbf{h}_t^0$ is defined as representation vector $\mathbf{x}_t$ of the t-th input word w_t^s. Then, $\mathbf{h}_t^L$ can encode the processed historical information as context vectors. The context vector representation for token w_t^s is denoted by

$$\mathbf{h}_{context}(w_t^s) = \mathbf{h}_{t-1}^L. \tag{1}$$

One can easily extend this unidirectional LMs into bidirectional LMs for the estimation of word embedding vectors. Bidirectional LSTMs (BiLSTMs) are designed by implementing both forward and backward LSTMs in parallel at each layer. BERT is basically a Transformer-based bidirectional model without the backward masking in GPT.

In Stage 1, a deep LSTM, a deep BiLSTM, a GPT, or a BERT LM is trained to minimize the negative log-likelihood (*NLL*) of the training sequence. As a result, one can obtain the implicit and explicit features for the words, i.e., optimized context vector $\mathbf{h}_{context}$ and single-sense word embeddings $\mathbf{X}$.

3.2. Stage 2: Sense labeling via unsupervised clustering

Stage 2 is based on the philosophy of the context clustering method used in the word sense induction task (Schütze, 1998).

As shown in Fig. 1(b), context vectors of a specific word are grouped according to similarity through unsupervised clustering. We adopt the k-means clustering algorithm, which is the most intuitive and fastest method (Lloyd, 1982).

When a word $v \in V_s$ occurs R_v times in the corpus, the set H_v of context vectors with a word v is denoted by $H_v = \{\mathbf{h}_{context}(v(1))..., \mathbf{h}_{context}(v(R_v))\}$. The number of context vectors in a word is the same as the number of times the word appears in the corpus.

Given an M-dimensional set of R_v data points (i.e., context vectors) for a word v in $H_v \in \mathbb{R}^{M\times R_v}$, the algorithm divides these points into $K_v (\leq R_v)$ clusters.

After clustering convergence, the sense of word v for each occurrence in the training corpus is assigned to the cluster with the closest centroid $\boldsymbol{\mu}_k^v$ $(k \leq K_v)$ in terms of the Euclidean distance as

$$z_v(r) = \arg\min_{k=1,\ldots,K_v} \|\mathbf{h}_{context}(v(r)) - \boldsymbol{\mu}_k^v\|, \quad (1 \leq r \leq R_v), \tag{2}$$

where $\mathbf{h}_{context}(v(r))$ is the r-th context vector of word v, which means that each word v is divided into K_v tokens for multi-sense LM learning in the next stage. Note that this clustering process proceeds separately for each word.

In our experiments, we consider two options to set the number of clusters (senses) K_v, i.e., a fixed number and different numbers of clusters for each word. For the latter, we estimated the number of clusters with the *gap statistic method* (Tibshirani, Walther, & Hastie, 2001). A detailed analysis on this estimation of cluster number is presented in Section 5.1.3.

3.3. Stage 3: Multi-sense LM learning

Although the cluster means in Stage 2 were used for multi-sense embedding vectors in previous works (Li & Jurafsky, 2015), for improved performance we only use the multi-sense labels and re-train LMs in Stage 3.

Stage 3 is very similar to Stage 1 with only one important difference. The input and output tokens in Stage 1 are single-sense words, while those in Stage 3 are multi-sense tokens obtained from Stage 2, which is equivalent to increasing the vocabulary size from $|V^s|$ to $|V^m| = \sum_{v\in V_s} K_v$. Therefore, multi-sense tokens w^m from the same word w^s now learn different representations to become a multi-sense LM.

3.4. Performance metrics for the language modeling task

Perplexity (*PPL*) has been widely used as a performance metric of LMs. Since *PPL* is based on the likelihood of a sentence, it is sensitive to the number of words in the corpus. However, the single-sense and multi-sense LMs from the same corpus result in different vocabulary sizes, and we propose a new metric, i.e., the unigram-normalized *PPL* (*PPLu*).

The *PPL* of an LM is defined as the length-normalized inverse geometric average probability of word sequences and estimated by

$$PPL = P(w_1, \ldots, w_T)^{-\frac{1}{T}} = \left(\prod_{t=1}^{T} P(w_t|w_{1:t-1})\right)^{-\frac{1}{T}}, \tag{3}$$

where T is the length of a sequence, and w_t is the word at time t in a sentence. An LM that achieves a higher probability for word sequences obtains a lower *PPL* value.

This *PPL* metric is suitable for comparing LMs with the same vocabulary. However, *PPL* may not be suitable for comparing LMs with different vocabulary sizes. A larger vocabulary size tends to result in lower word and sentence probabilities and thus in higher *PPL*.

To overcome the limitation of *PPL*, we adopt the basic idea of normalizing the word probability with respect to a quantity containing the vocabulary size. The proposed *PPLu* is defined as

$$PPLu = \left(\prod_{t=1}^{T} \frac{P(w_t|w_{1:t-1})}{P_{uni}(w_t)}\right)^{-\frac{1}{T}} = \frac{P(w_1, w_2, \ldots, w_T)^{-\frac{1}{T}}}{\left(\prod_{t=1}^{T} P_{uni}(w_t)\right)^{-\frac{1}{T}}}, \tag{4}$$

where T is the length of a sequence and the unigram probability is computed as

$$P_{uni}(w_t = v_k) = \frac{Count(v_k)}{\sum_{k'=1}^{|V|} Count(v_{k'})} = P(w_t), \tag{5}$$

where $Count(v_k)$ is the number of occurrences of word v_k in the corpus. This metric shows the likelihood improvement of a context-dependent LM from the simple unigram LM without context information, and therefore enables us to evaluate the effectiveness of a context-dependent LM.

From (4), we may interpret the *PPLu* in terms of the Kullback–Leibler (KL) divergence and mutual information (MI) as

$$\begin{aligned}\log PPLu &= -\frac{1}{T}\sum_{t=1}^{T} \log \frac{P(w_t|w_{1:t-1})}{P(w_t)} \\ &= -KL(P(w_t|w_{1:t-1}) \parallel P(w_t)) \\ &= -\frac{1}{T}\sum_{t=1}^{T} \log \frac{P(w_t, w_{1:t-1})}{P(w_t)P(w_{1:t-1})} = -I(w_t; w_{1:t-1}).\end{aligned} \tag{6}$$

$KL(a \parallel b)$ and $I(a; b)$ denote the KL divergence of a from b and MI between a and b, respectively. Utilizing the context information $w_{1:t-1}$, high-performance LMs make $P(w_t|w_{1:t-1})$ diverge from $P(w_t)$ and increase the amount of shared information between w_t and $w_{1:t-1}$. Therefore, *PPLu* becomes smaller for better LMs.

Additionally, we present how the proposed *PPLu* is affected by the number of words in the corpus. We show a very simple example in which a word v_{ab} is used confusingly for two words, v_a and v_b. If a sentence does not include v_{ab}, the split does not have an effect on either *PPL* or *PPLu*. If a sentence includes either v_a or v_b as the τth word, then

$$\begin{aligned}\log PPLu = -\frac{1}{T}[&\log P(w_1, \ldots, w_\tau = (v_a \text{ or } v_b), \ldots, w_T) \\ &-\{\sum_{t\neq\tau}^{T} \log P(w_t) + \log P(w_\tau = (v_a \text{ or } v_b))\}].\end{aligned} \tag{7}$$

If word v_{ab} actually has only one meaning and we randomly assign v_a or v_b instead of v_{ab} with probabilities β and (1-β), respectively, then

$$\begin{aligned}&P(v_a) = \beta P(v_{ab}), P(v_b) = (1-\beta)P(v_{ab}), \\ &P(w_1, \ldots, w_\tau = v_a, \ldots, w_T) = \beta P(w_1, \ldots, w_\tau = v_{ab}, \ldots, w_T), \\ &P(w_1, \ldots, w_\tau = v_b, \ldots, w_T) = (1-\beta)P(w_1, \ldots, w_\tau = v_{ab}, \ldots, w_T).\end{aligned} \tag{8}$$

Therefore, for a sentence with v_a,

$$\begin{aligned}\log PPLu = -\frac{1}{T}[&\log\beta + \log P(w_1, \ldots, w_\tau = v_{ab}, \ldots, w_T) \\ &-\{\sum_{t\neq\tau}^{T} \log P(w_t) + \log\beta + \log P(w_\tau = v_{ab})\}],\end{aligned} \tag{9}$$

which is exactly the same as the *PPLu* value of the original case without splitting. It is worth noting that the split makes $\log PPL$

increase by $-\log\beta/T$. Moreover, the same is true for sentences with v_b. In this case, although the vocabulary size is increased by 1, *PPLu* is invariant with the split of a word. Since the two words v_a and v_b are semantically identical, this simple split should not result in a difference in LM performance.

If the word v_{ab} has multiple senses and is divided into multiple tokens based on meaning, then the split is no longer random, and β, the ratio of $P(\ldots, w_i = v_a, \ldots)$ to $P(\ldots, w_i = v_{ab}, \ldots)$, is no longer a fixed constant and depends upon the other words in the sentence. Therefore, *PPLu* values with and without the split become different. Since *PPLu* is invariant with the vocabulary size and is only a function of the mutual information between w_t and $w_{1:t-1}$, the difference in value of *PPLu* purely comes from LM performance. It is straightforward to extend this analysis to multiple splits and many multi-sense words.

4. Data and experimental settings

4.1. Datasets

To validate our method in LM performance, we used three benchmark language modeling datasets, i.e., the SemCor (Miller, Leacock, Tengi, & Bunker, 1993), PTB (Marcus, Marcinkiewicz, & Santorini, 1993), and Text8 (Mikolov, Joulin, Chopra, Mathieu, & Ranzato, 2014) datasets. Also, eight NLP tasks of the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) and the three tasks of the SuperGLUE (Wang et al., 2019) benchmark were evaluated.

The three language modeling datasets differ in their number of tokens and vocabulary sizes, as summarized in Table 1. The SemCor dataset contains text data extracted from the Brown Corpus and their sense annotation from the New Oxford American Dictionary (NOAD) or WordNet inventory. The SemCor dataset was selected due to its availability of human-annotated multi-sense labels. The PTB dataset is a subset of the Wall Street Journal (WSJ) dataset, which contains news articles on business-oriented topics. The Text8 dataset is derived from a Wikipedia dump and contains much broader topics compared to the PTB dataset.

The GLUE benchmark includes nine general NLP tasks, i.e., two single-sentence classification tasks, such as CoLA (Warstadt, Singh, & Bowman, 2019) and SST-2 (Socher et al., 2013), and seven sentence-pair classification tasks, such as MRPC (Dolan & Brockett, 2005), QQP, STS-B (Cer, Diab, Agirre, Lopez-Gazpio, & Specia, 2017), MNLI (Williams, Nangia, & Bowman, 2018), QNLI (Rajpurkar, Zhang, Lopyrev, & Liang, 2016), RTE (Dagan, Glickman, & Magnini, 2005), and WNLI (White, Rastogi, Duh, & Van Durme, 2017). We evaluated our models on eight tasks (i.e., SST-2, MRPC, QQP, MNLI, QNLI, RTE, CoLA, and STS-B). For the WNLI task, no model in the literature exceeded the most frequent class guessing (56.3%), and it may not be worthwhile to report results in this paper. Additionally, we evaluated our models for three tasks in SuperGLUE benchmark. All of these tasks are sentence-pair classification tasks. Specifically, BoolQ (Clark et al., 2019) and MultiRC (Khashabi, Chaturvedi, Roth, Upadhyay, & Roth, 2018) are a QA-based task, and WiC (Pilehvar & Camacho-Collados, 2019) is a word sense disambiguation task. The other tasks are either based on tiny datasets or almost the same as that in GLUE.

4.2. Selection of multi-sense words

Stage 2 needs to identify which words may have multiple senses. However, such ground-truth annotations do not exist in most text datasets, including the PTB and Text8 datasets, and human annotation requires considerable effort. For computational efficiency, we removed two types of words from the multi-sense word candidates. First, we excluded *stop words* (i.e., words with very little meaning such as "is" and "are"). In employing the stop word list in the NLTK toolkit (Loper & Bird, 2002), we removed the stop words that accounted for approximately 50% of the overall data in each dataset (e.g., corresponding to 132 stop words for the PTB dataset). Second, we excluded *rarely appearing words* for effective clustering with enough samples. After these exclusions, the remaining 1854 words for the PTB dataset, 1755 words for the SemCor dataset, and 11,147 words for the Text8 dataset were used as possible multi-sense words. Note that Stages 1 and 3 still utilized all the words in the corpus.

Table 1
Data statistics for language modeling datasets. $|V|$ is the vocabulary size, and $|T|$ is the total number of tokens in the dataset. The SemCor dataset has annotated sense labels by human experts for all data.

Dataset	$\|V\|$	$\|T\|$
SemCor all-words (SemCor) (Miller et al., 1993)	≈10 k	0.43 M
Penn Treebank (PTB) (Marcus et al., 1993)	10 k	1 M
Text8 (Mikolov et al., 2014)	≈44 k	17 M

4.3. Network architecture

In Stages 1 and 3, LMs with multi-layer LSTM or Transformer modules are incorporated. For some implementations the word embedding dimension is specified with "-W" prefix in the model name, e.g., LSTM–W650. The dimension of hidden representation is the same as that of word embedding. For each dataset, we examined various architectures and hyperparameters, i.e., embedding dimensions and sense numbers.

5. Experimental results

In this section, we perform the language modeling task on the SemCor, PTB, and Text8 dataset, and the NLU tasks on the GLUE and SuperGLUE benchmarks.

For all experimental results, we conduct Welch's t-test to identify the statistically significant differences in the results between the models with single-sense tokens and multi-sense tokens for the same network. We mark star(*) when significant difference ($P < 0.05$) occurred.

5.1. Performance on the language modeling task

For each dataset, we report the results of two LSTMs with different word embedding dimensions. Following Kim et al. (2016), we train LSTM with two hidden layers via truncated backpropagation-through-time. We use stochastic gradient descent with backpropagation for 35 time steps. Additionally, for the Text8 dataset, we add Transformer-based unidirectional LM, i.e., GPT-2, with a similar number of parameters. We train a 12-layer network with 12 attention heads. Following Radford et al. (2019), we used Adam optimizer with a 0.0001 max learning rates.

5.1.1. Results on a fixed number of senses for all words

The SemCor dataset is small but has human-annotated labels for multiple senses. Because no previous LM method has been evaluated on the SemCor dataset, we compared our LSTM-based methods of several different sense numbers, i.e., all single sense, all 5 senses, exact number of senses for each word, and exact sense labels. The third and fourth are possible due to the existence of the ground-truth multi-sense labels. Here, we construct small models due to the size of the SemCor dataset and, as expected, the model with 400 embedding dimension results in much better performance than that with 100 embedding dimension.

Table 2
Performance on the SemCor dataset. The bold font indicates the best unigram-normalized perplexity (*PPLu*) for the same model. Here, "Exact Numbers" and "Exact Labels" denote the MSLMs with the exact number of senses (K) for each word and the exact sense labels in the dataset, respectively. Star(*) indicates significant difference ($P < 0.05$) between single-sense and multi-sense models with the same network. *PPL* is not designed to compare LM performance between different vocabulary sizes, and is shown for reference only.

Model	Number of senses	Metrics	
		PPL	*PPLu*
LSTM–W100	1 for all	116.5	0.2491
	5 for all	142.9	0.2137*
	Exact Numbers	137.8	**0.1934***
	Exact Labels	142.2	0.1969*
LSTM–W400	1 for all	102.6	0.2195
	5 for all	134.1	0.2005*
	Exact Numbers	130.1	**0.1824***
	Exact Labels	131.7	0.1848*

It is worth noting that the multi-sense LMs inherently use a larger vocabulary size than the single-sense LMs, and thus result in higher *PPL* values. The *PPL* values of multi-sense LMs with the exact labeled senses are still much larger than those of single-sense LMs, which proves that *PPL* is not a good measure for comparing LM performance between two datasets with different vocabulary sizes.

In terms of the proposed *PPLu* measure, as expected, the multi-sense LMs show much better performance than the single-sense LMs with the same embedding dimension. Compared to the single-sense LMs the multi-sense LMs with 5 sense clusters for all words reduce *PPLu* by 14.2% (from 0.2491 to 0.2137) and 8.7% (from 0.2195 to 0.2005) for the word embedding dimensions of 100 and 400, respectively. Then, the multi-sense LMs using the exact number of senses for each word further reduce *PPLu* by another 9.5% (from 0.2137 to 0.1934) and 9.0% (from 0.2005 to 0.1824), respectively. The LMs with exact sense labels are expected to be the best, but are slightly worse than those from exact sense numbers. The difference is within the error margin of the model and small dataset, which shows that our clustering algorithm results in almost optimal clusters. This result implies that the precise estimation of the number of senses can improve LM performance.

Additionally, Table 2 shows that the *PPL*/*PPLu* value are quite similar for LMs with different embedding dimensions, but show big differences with different sense numbers. This finding was attributed to the definition of *PPLu*, which is a normalized *PPL* using unigram probabilities. By definition, the models that shared the same unigram probabilities showed the same ratio value of $PPL/PPLu = \left(\prod_{t=1}^{T} P_{uni}(w_t)\right)^{-\frac{1}{T}}$. Because the settings under different numbers of senses (e.g., 1 for all vs. 5 for all) used different word vocabularies and thus different *PPL*/*PPLu*.

We conduct Welch's t-test to identify the statistically significant differences in the results of single-sense and multi-sense LMs with the same network architecture. Specifically, the *PPL* (or *PPLu*) scores of the test sentences were used as samples for the t-test (i.e., sample no. = 1,641 for each model), and the null hypothesis assumed that the scores of the two models exhibit no meaningful difference. By rejecting the null hypothesis with $p = 0.05$, we confirm that the following observations are statistically significant: (i) multi-sense LMs show superior performance to single-sense LMs. (ii) multi-sense LMs with ground-truth sense information (i.e., exact sense numbers and exact sense labels) show better *PPLu* than the LMs with a fixed number of senses. (iii) There is no significant difference between the multi-sense LMs with ground-truth sense numbers and sense labels.

Table 3
Performance of single-sense (SS) and multi-sense (MS) LMs on the PTB dataset. The metrics *PPL* and *PPLu* correspond to perplexity and unigram-normalized perplexity, respectively. The bold font indicates the best *PPLu* for all experiments. Star(*) indicates significant difference ($P < 0.05$) between SS and MS models with the same network.

Model	SS/MS (Number of sense(s))	Metrics	
		PPL	*PPLu*
LSTM–W200	SS (K = 1) (Kim et al., 2016)	97.6	0.1527
	MS (K = 9)	129.2	0.1051*
LSTM–W650	SS (K = 1) (Kim et al., 2016)	85.4	0.1336
	MS (K = 9)	124.8	**0.1015***
LSTM-variants	with augmented loss (Inan et al., 2016)	73.2	0.1145
	with continuous cache pointer (Grave et al., 2016)	72.1	0.1128

Table 4
Performance of single-sense (SS) and multi-sense (MS) LMs on the Text8 dataset. The best results among all experiments are shown in bold fonts. Star(*) indicates significant difference ($P < 0.05$) between SS and MS models with the same network.

Model	SS/MS (Number of senses(s))	Metrics	
		PPL	*PPLu*
LSTM–W300	SS (K = 1)	159.4	0.1138
	MS (K = 9)	199.9	0.0503*
LSTM–W650	SS (K = 1)	174.8	0.0898
	MS (K = 9)	137.8	0.0440*
GPT2–W600	SS (K = 1)	98.9	0.0706
	MS (K = 9)	153.4	0.0386*
	MS (K = 9) with GPT2 at Stage 1	150.0	**0.0360***

To compare our results with existing published results, we also applied our methods to the PTB dataset and the Text8 dataset. Some *PPL* values were imported from the published literature, and *PPLu* values were evaluated from *PPL* and unigram probabilities.

Table 3 shows the comparison on the PTB dataset. Our results are obtained with LSTM LMs with 200 and 650 word embedding dimensions. Although there exist several published results that do not use LSTM, e.g., Kneser Ney smoothing, RNN (Mikolov et al., 2011), CNN (Wang, Lu, Li, Jiang, & Liu, 2015), and Sum–Product Net (Cheng et al., 2014), they perform significantly worse than LSTM. Therefore, we use the LSTM-based LMs as our single-sense LM. In terms of the proposed *PPLu*, our multi-sense LMs with 9 senses performed much better than the single-sense LMs. Notably, the small-size multi-sense LM outperforms both the large-size single-sense LMs. Additionally, the improvements from single-sense to multi-sense LMs on the same network architectures are significant, i.e., a 31.2% reduction from 0.1527 to 0.1051 and a 24.0% from 0.1336 to 0.1015 for the 200 and 650 embedding dimensions, respectively.

Table 4 shows the comparison on the Text8 dataset, which is larger than the PTB and SemCor datasets in terms of vocabulary size and number of words. For our multi-sense LMs, we employ the same architectures used on the PTB dataset to test their scalability on medium-scale data. With this much larger Text8 dataset, we are also able to incorporate the GPT-2 LMs in Stage 3.

Again, we observed a similar trend in the results. In terms of *PPLu*, even the small multi-sense LSTM LM with 300 embedding dimension showed better performance than single-sense LMs with larger embedding dimension including GPT-2 and the neural cache model (Grave et al., 2016) (*PPL* is 99.9 and *PPLu* is 0.0713). In addition, multi-sense GPT-2 models show significantly better performance than all the others. For examples, the *PPLu* values

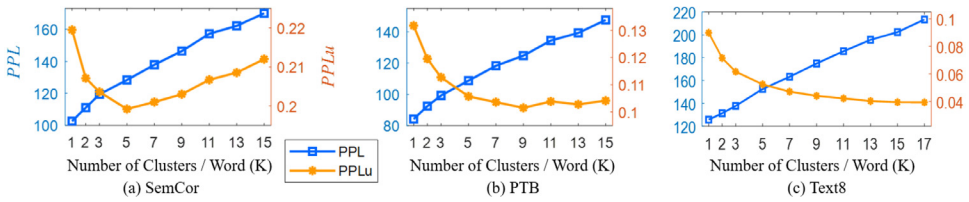

Fig. 2. Performance of our LSTM-based multi-sense LMs for a different number of senses (clusters) on the (a) SemCor, (b) PTB, and (c) Text8 datasets. The results obtained with K = 1 correspond to the single-sense LSTM-LMs. The results obtained with K > 1 correspond to our multi-sense LMs based on LSTM–W400 or LSTM–W650.

of multi-sense LMs become much smaller than those of single-sense LMs by 55.8% (from 0.1138 to 0.0503) for LSTM–W300, by 51.0% (from 0.0898 to 0.0440) for LSTM–W650, and by 45.3% (from 0.0706 to 0.0386) for GPT2–W600.

Also, we found that the LM model in Stage 3 is more important than that in Stage 1. While the other LMs in Stage 1 are based on LSTM, that of the bottom row in Table 4 is implemented with GPT-2. Therefore, the difference between *PPLu* values at the first and second rows from the bottom come solely from the different LM models in Stage 1. By changing LM model from LSTM to GPT-2 in Stage 1 with the same GPT-2 model in Stage 3, we obtain only 6.7% reduction of *PPLu* (from 0.0386 to 0.0360). On the other hand, with the same LSTM-based LM in Stage 1, the change of LM from LSTM to GPT-2 in Stage 3 resulted in 12.3% (from 0.0440 to 0.0386) reduction of *PPLu* values.

5.1.2. Effects of the number of senses on LM performance

The number of clusters (i.e., the number of senses per word), K, is an important hyperparameter for the *k*-means clustering in Stage 2, which affects multi-sense LM performance in Stage 3. Here, we investigate how the number of clusters, K, affects performance. We show the results from the same LSTM models for both Stages 1 and 3. For the case with K = 1 (which is identical to single-sense LM), the results of LSTM models in Stage 1 are reported. For each case with K>1, we accordingly form new vocabularies and train our LSTM-based multi-sense LMs in Stage 3. We perform this experiment for each dataset.

Fig. 2 shows the effect of varying K, and we can make two observations, i.e., (i) data characteristics affect the proper number of clusters, and (ii) *PPL* and *PPLu* show different trends. Regarding (i), the number of clusters for the best *PPLu* depends upon dataset (i.e., K = 5 for SemCor, K = 9 for PTB, and K ≥ 17 for Text8). The Text8 dataset is much larger and has broader topics than the PTB and SemCor datasets, which may lead to a larger optimal number of clusters. Therefore, it is necessary to find the proper number of clusters for each corpus.

Regarding (ii), *PPL* values increase monotonically as the number of clusters increases (i.e., the vocabulary size increases). This result again supports that *PPL* is not suitable for comparing LMs with different vocabulary sizes. On the other hand, as the number of clusters increase, *PPLu* values decrease initially and eventually converge to fixed values or even increase. For small SemCor dataset the large numbers of senses may result in overfitting, while the large number of clusters may not harm much for the largest Text8 dataset. Provided enough training data were available, the extra number of clusters may just result in splitting a sense into several clusters, which does not change *PPLu* values as demonstrated in Section 3.4.

5.1.3. Results with the estimated number of senses for each word

In Table 5, we show both the multi-sense LM performance in Stage 3 and the clustering performance in Stage 2. We also employ the *gap statistics methods* (Tibshirani et al., 2001) to estimate the number of clusters for each word. Specifically, given the learned context embedding vectors of each word occurrence in Stage 1, we perform *k*-means clustering by varying the number of clusters, K, in a range of $1 \leq K \leq K_{max}$. Then, we select the best value of K by evaluating the KL divergence of each cluster from uniform random distribution. It assumes that uniformly distributed clusters are not good.

Table 5
Experimental results on the SemCor, PTB, and Text8 datasets with the estimated optimal number of senses for each word. Bold fonts indicate the best performance on each dataset. For the SemCor dataset with annotated sense labels, clustering results in Stage 2 are evaluated with the Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI). Star(*) indicates significant ($P < 0.05$) difference between single-sense and multi-sense models with the same network.

Datasets	Number of senses	Metrics			
		PPL	*PPLu*	ARI	NMI
SemCor	Fixed K = 1	102.6	0.2195		
	Fixed K = 5	134.1	0.2005*	7.1	20.7
	Gap statistics	135.1	0.1970*	7.1	24.2
	Exact Numbers	130.1	**0.1824***	10.2	34.5
PTB	Fixed K = 1	85.4	0.1336		
	Fixed K = 9	124.8	**0.1015***	-	-
	Gap statistics	124.5	0.1029*		
Text8	Fixed K = 1	98.9	0.0706		
	Fixed K = 9	174.8	**0.0440***	-	-
	Gap statistics	175.3	0.0475*		

By estimating the number of senses for each word with the gap statistics method, the *PPLu* values of multi-sense LMs became smaller than or similar to those of the exact-numbers. For the PTB dataset, *PPLu* value with the gap statistics method is comparable to that of fixed sense number (K = 9). However, for the largest Text8 dataset, the gap statistics method results in slightly poorer performance.

Fig. 3 shows the distributions of the estimated number of clusters in three datasets. We set the maximum number of senses (K_{max}) for each dataset to 10, which is the same as the maximum number of senses for SemCor annotation. The average numbers of clusters for the SemCor, PTB, and Text8, are 5.9, 7.9, and 9.2, respectively. Results of the gap statistics method result in a similar number of clusters as those of the best *PPLu*. On the other hand the numbers of words with larger number of clusters (K ≥ 7) are still increasing, which may give us a hint for the needs of even larger number of clusters. Actually, with the largest word size in the Text8 dataset, there may be words with quite large number of senses. Note that the use of the same number of senses requires a hyperparameter search to find an optimal K. In contrast, the use of an estimation method does not require such a time-consuming search and is a more natural approach to processing words with a different number of senses.

Additionally, we include a quantitative evaluation of the clustering results of Stage 2 with two additional evaluation metrics, i.e., the ARI (Hubert & Arabie, 1985) and NMI (Kvalseth, 1987). Both ARI and NMI have an expected value of "0" for independent

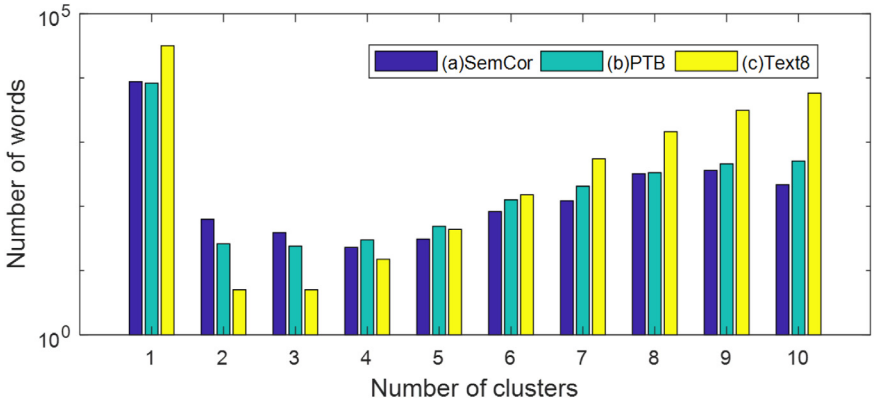

Fig. 3. Distribution of the number of clusters derived from the *gap statistics* methods on the (a) SemCor (blue), (b) PTB (green), and (c) Text8 (yellow) datasets. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

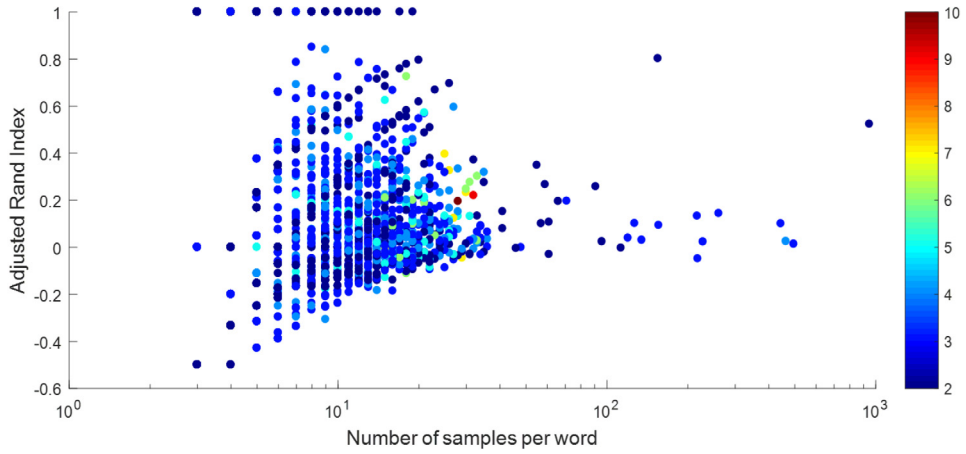

Fig. 4. Scatterplot of the number of samples per word and the ARI.

clusters and "1" for identical clusters. To evaluate the clustering performance, the ground-truth clustering labels are needed, and results on the SemCor dataset only are reported here. Following the common practices, we multiply the ARI and NMI values by 100. As shown in Table 5, both the ARI and NMI values of the models using exact and estimated number of clusters become higher than those of models using a fixed number of clusters. Moreover, the NMI value of the model using the estimated number of clusters has higher than those of the fixed number of clusters ($K = 5$). These results are in agreement with the tendency of *PPLu*.

Although the ARI and NMI values with exact sense numbers are expected to be the closest to the ground-truth clusters, they are much smaller than those of exact labels, i.e., ARI=NMI=100, due to the small number of samples for each word. Fig. 4 shows a scatter plot of words in (the number of samples per word) and (the ARI value) space. One dot represents a word, and the color of the dot represents the number of senses in the word. We observed that the variance in the ARI value increases as the number of samples per word decreases. This phenomenon is especially critical for the ARI measure, which is based on selecting 2 samples in the same cluster. Since 50% of the words in the SemCor dataset have less than 10 samples per word, the ARI values are much smaller than expected. Even with the exact number of clusters, the clustering becomes noisy with a small number of samples. If a sufficient number of samples is provided for each word, the overall average value of ARI can be expected to increase.

5.1.4. Nearest-neighbor words with multiple senses

In Table 6, we further present our qualitative results on the SemCor dataset in terms of the nearest-neighbor words. For each query word with a single-sense LM, we found top-3 nearest words based on Euclidean distance in the word embedding space. However, due to the multi-sense property, the nearest-neighbor words do not show any consistency. Also, for each query word with a multi-sense LM, we found top-2 senses and their top-3 nearest neighbor words. Both the exact numbers of clusters (denoted as "Exact Sense Numbers") and the ground-truth sense labels (denoted as "Exact Sense Labels") are compared. For the query "like", its two sense embedding vectors show two distinct neighbors, one as a verb and the other as a preposition. The nearest neighbors of the query "most" also show distinct usages as determiner and adverb. However, the word "may" is used as an auxiliary verb in almost all training samples, and its other usages (as the fifth month of the year) contribute little to the cluster learning. Therefore, all sense clusters may result in similar embedding vectors and have the same neighbors. All these phenomena are related to the fact that the multi-sense clustering works well.

Similarly Table 7 qualitatively compares the nearest-neighbor words of single-sense and multi-sense representations on the Text8 dataset. Again, the single-sense and multi-sense representations lead to similar but slightly different tendency. For example, the query "may" in the multi-sense LM now has two distinct neighbor words, one for the auxiliary verb and the other for a month of the year. The larger data size may contribute to this better clustering results.

Table 6
Top-3 nearest-neighbor words of the single-sense and the multi-sense representations from LSTM–W400 in the SemCor dataset.

Query	*like*		*most*		*then*		*may*	
Single-sense	*including* *met* *provides*		*some* *many* *five*		*finally* *otherwise* *already*		*might* *shall* *should*	
Exact sense numbers	*wanted* *want* *used*	*by* *about* *including*	*many* *five* *seven*	*relatively* *perfectly* *extremely*	*sometimes* *merely* *once*	*night* *moment* *here*	*should* *must* *would*	*would* *might* *will*
Exact sense labels	*need* *want* *wanted*	*including* *by* *against*	*many* *some* *five*	*too* *quite* *become*	*recently* *sometimes* *suddenly*	*years* *night* *moment*	*shall* *must* *could*	*might* *should* *simply*

Table 7
Top-4 nearest-neighbor words of the single-sense and the multi-sense representations from LSTM–W650 in the Text8 dataset.

Query	*may*		*banks*		*fox*		*rock*	
Single-sense	*might* *can* *to* *could*		*generators* *vicinity* *bubbles* *capitalists*		*lancaster* *richardson* *turner* *robertson*		*jam* *bluegrass* *funk* *ceramic*	
Multi-sense	*june* *august* *april* *december*	*shall* *might* *cannot* *otherwise*	*shores* *corners* *tributaries* *slopes*	*corporations* *businesses* *firms* *agencies*	*nbc* *hbo* *cbs* *aol*	*panda* *dodo* *kangaroo* *dish*	*wind* *springs* *cave* *sand*	*pop* *punk* *blues* *techno*

5.2. Performance on the NLP benchmarks

We compare performance on several downstream NLP tasks based on the GLUE and SuperGLUE benchmarks.

5.2.1. Setup

To ensure a fair comparison of the representations, all baselines were trained from scratch on the Text8 dataset (Section 5.2.2) with the dimensionality of the embeddings set to 300 (default). The same model is used for both the single-sense and multi-sense LMs in Stages 1 and 3.

For the LSTM-based LMs, the pretrained embedding vectors were used as inputs to the BiLSTM encoder, which was trained for each downstream task. In the single-sentence classification task, SST-2, similar to Wang et al. (2018), we used a two-layer BiLSTM with 1,200-dimension embedding vectors each for forward and backward directions with max pooling as the sentence encoder. The output of this sentence encoder is fed to the task classifier. In the sentence-pair classification tasks, i.e., MRPC, QQP, MNLI, QNLI, RTE, BoolQ, WiC, and MultiRC, following Mou et al. (2015), we independently encoded each of the sentences h_1 and h_2 and passed the feature vectors $[h_1; h_2; h_1 \cdot h_2; h_1 - h_2]$ to a task classifier.

For the Transformer-based models, the parameters of the pretrained models were fine-tuned for each downstream task. Due to the memory capacity of our GPU system we used 12-layer networks with 12 attention heads and 128 time steps. Following the original papers, we used Adam adaptive learning rates with 0.0001 as a maximum learning rate. The final hidden state vector corresponding to the special classification token for "Classification" was used as a sentence representation for classification tasks.

Our baseline models are described below.

- CBoW and SG+CRP (Li & Jurafsky, 2015): These are n-gram methods and included only for reference purpose. The CBoW is a single-sense model, while the SG+CRP utilizes skip-gram (SG) embedding learning and unsupervised clustering of multiple senses by Chinese Restaurant Process (CRP). Each sense token is assigned by using a greedy search algorithm. The codes for SG+CRP have been provided at https://github.com/jiweil/mutli-sense-embedding.
- LSTM and BiLSTM: Follows to Kim et al. (2016), the LSTM is a two-layer LSTM LM. Word embeddings of 300 dimensions are used as the input to the sentence encoder. The BiLSTM is a two-layer BiLSTM LM as Melamud et al. (2016). Output embeddings of 300 dimensions are used as the pretrained input to the sentence encoder.
- GPT2 and BERT: We follow the implementation of GPT-2 described in Radford et al. (2019) and BERT in Devlin et al. (2019) with a word-sense embedding dimension of 600. To match the vocabulary size of the other baseline models, we do not use subword embedding in GPT-2 and BERT and use a word-sense embedding only. The codes for GPT-2 and BERT have been provided at https://github.com/huggingface/transformers.
- XLNet: XLNet is a variant of Transformer-based LMs with bidirectional contexts. We follow the training procedures used in Yang et al. (2019) with a word/sense embedding dimension of 600.

5.2.2. Results on the GLUE benchmark

Table 8 shows performance on the GLUE benchmark. For the same network architecture and parameters, the performance of multi-sense LMs is better than or similar to that of single-sense LMs for all tasks. Especially, multi-sense GPT-2 and BERT achieved significantly higher scores on the textual entailment tasks such as MNLI and QNLI. However, the multi-sense token method does not help much in improving performance on the RTE task. Also, GPT-2 and BERT result in similar to each other, but better than LSTM-based models on several tasks and on average. On the other hand the multi-sense LSTM LM achieved the highest accuracy (77.7%) on the MRPC task, which evaluates the semantic similarity between two sentences. These results confirm that our sense-level tokens can help improving the performance of various LM models including LSTMs and Transformers. Similarly, another unsupervised multi-sense model, SG+CRP, showed higher scores than the single-sense CBoW method. However, these n-gram models are inferior to the LSTM and Transformer-based LMs.

Table 8
Performance on the GLUE benchmark. All models are pretrained on the Text8 dataset. For CoLA and STS-B, we report the Matthews correlation coefficients and the averaged values of the correlation coefficients, respectively. For all other tasks an accuracy is reported for each task and also their average value. Performance of single-sense CBoW reported in Wang et al. (2019) and multi-sense SG with CRP in Li and Jurafsky (2015) is reported for reference only. Bold fonts indicate the best performance for each task. Except the average accuracy, a star(*) indicates statistically significant difference ($P < 0.05$) between single-sense and multi-sense models with the same network.

Model	SS/MS	Avg.	SST-2	MRPC	QQP	MNLI	QNLI	RTE	CoLA	STS-B
CBoW	SS	69.1	80.0	73.4	79.1	56.0	72.1	54.1	0.0	60.0
SG+CRP	MS	72.7	78.3	74.3	84.6	67.1	73.2	58.5	8.6	67.9
LSTM	SS	73.8	80.2	75.0	85.3	67.9	74.7	**59.9**	15.9	69.8
	MS	74.6	81.2*	**77.7***	86.2*	67.5*	75.0*	**59.9**	**20.8***	69.4*
BiLSTM	SS	73.5	79.4	75.7	84.9	67.5	75.1	58.1	18.0	67.7
	MS	74.3	81.7*	76.0	86.1*	68.1*	74.4*	59.6*	18.1*	**71.9***
GPT2	SS	73.2	85.9	71.8	82.7	64.7	75.3	58.8	18.4	41.3
	MS	76.2	**86.7***	74.0*	**86.7***	72.8*	77.9*	58.8	17.3*	69.6*
BERT	SS	72.7	85.0	72.1	82.7	63.8	74.7	58.1	16.6	30.9
	MS	**76.4**	85.1	73.3*	86.4*	**73.3***	**80.4***	**59.9**	17.5*	69.3*

Table 9
Performance of single-sense (SS) and multi-sense (MS) models on three tasks in the SuperGLUE benchmark. For BoolQ and WiC, we report accuracies, while we report F1 on all answer-options for MultiRC. In the Avg. column, we report the average value of all tasks. Performance of CBoW was reported in Wang et al. (2019). Bold fonts indicate the best performance for each task set. A star(*) indicates significant difference ($P < 0.05$) between SS and MS models with the same network.

Model	SS/MS	Avg.	BoolQ	WiC	MultiRC
CBoW	SS	46.0	62.4	55.3	20.3
SG+CRP	MS	58.7	64.7	59.7	51.6
LSTM	SS	61.7	69.1	61.4	54.6
	MS	62.3	68.8*	63.0*	55.2*
BiLSTM	SS	62.0	69.3	61.8	55.0
	MS	62.6	**69.4***	63.0*	55.5*
GPT2	SS	62.2	68.9	60.2	57.4
	MS	63.1	**69.4***	60.2	59.7*
BERT	SS	62.1	67.2	60.2	58.9
	MS	62.6	67.8*	60.2	**59.9***
XLNet	SS	63.7	69.0	62.5	59.5
	MS	**65.0**	69.0	**66.0***	**59.9***

5.2.3. Results on the SuperGLUE benchmark

Table 9 compares the performance of the single-sense and multi-sense LMs on the three tasks (i.e., BoolQ, WiC and MultiRC) in the SuperGLUE benchmark. The performance of the SuperGLUE benchmark also shows similar tendency to that of the GLUE benchmark. For all LMs, the performance of multi-sense tokens shows better than or similar to that of single-sense tokens. The XLNet model results in the highest scores for average as well as WiC and MultiRC tasks with both single-sense and multi-sense tokens. For both single-sense and multi-sense tokens the BiLSTM shows the best score on the BoolQ. We observed that the SG-based multi-sense model showed poorer performance than the other LMs due to the difficulties of considering longer contexts. These results confirm that multi-sense tokens can help to improve the performance of various LMs significantly.

6. Conclusions

In this paper, we proposed a sense-aware framework that consists of three essential stages, i.e., *context representation learning stage*, *sense labeling stage*, and *multi-sense LM learning stage*, for introducing multi-sense word tokens. The proposed framework is the first to integrate the modeling of multi-sense words into various deep sequence neural LMs, which capture lengthy context information. By incorporating the multi-sense labeling process with unsupervised clustering methods, our multi-sense LMs successfully learn sense-specific representations. The unsupervised clustering stage is separated from the single-sense representation learning stage and also the fine-tuning stage with the learned multi-sense labels. Therefore, any state-of-the-art LMs can be easily adopted. Also, the number of senses for each word is estimated for near-optimal performance. This multi-sense clustering algorithm is implemented with purely unsupervised learning and applicable to almost all datasets. Because of the explicit assignment of different sense tokens to a multi-sense word, the LM task itself becomes much simpler for the same performance, and the resulting model is very light in terms of memory and computation requirements.

Furthermore, we proposed a new evaluation metric, *PPLu*, and provided both theoretical and experimental analyses. The new metric allows us to assess the performance difference between single-sense and multi-sense LMs with different vocabulary sizes.

By using three language modeling datasets, we demonstrated that the proposed multi-sense LMs significantly outperformed single-sense LMs with the same network architectures and parameters for all LSTM and Transformer-based models. Both unidirectional and bidirectional network architectures were implemented. The qualitative results of nearest-neighbor words confirmed the benefit of employing multi-sense representations in accurately capturing word meanings. Also, multi-sense LMs attained significantly better performance than single-sense LMs with the same architecture (LSTM, BiLSTM, GPT2, BERT, and XL-Net) and parameters for many downstream NLP tasks on the GLUE and SuperGLUE benchmarks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was jointly supported by the Industrial Strategic Technology Development Program (10072064, Development of Novel Artificial Intelligence Technologies to Assist Imaging Diagnosis of Pulmonary, Hepatic, and Cardiac Diseases and their Integration into Commercial Clinical PACS Platforms) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea) and Institute of Information & Communications Technology Planning & Evaluation (2016-0-00562, Emotional Intelligence Technology to Infer Human Emotion and Carry on Dialogue Accordingly) funded by the Ministry of Science and ICT (MSIT, Korea).

References

Aharoni, R., & Goldberg, Y. (2020). Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 7747–7763). Online: Association for Computational Linguistics.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *Proc. ICML* (pp. 173–182).

Ansell, A., Bravo-Marquez, F., & Pfahringer, B. (2021). PolyLM: Learning about polysemy through language modeling. arXiv:2101.10448.

Bartunov, S., Kondrashkin, D., Osokin, A., & Vetrov, D. (2016). Breaking sticks and ambiguities with adaptive skip-gram. In *Artificial intelligence and statistics* (pp. 130–138).

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *JMLR*, *3*(Feb), 1137–1155.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *The Journal of Machine Learning Research*, *3*, 993–1022.

Botha, J., & Blunsom, P. (2014). Compositional morphology for word representations and language modelling. In *Proc. ICML* (pp. 1899–1907).
Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., & Specia, L. (2017). Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. ArXiv preprint.
Cheng, W.-C., Kok, S., Pham, H. V., Chieu, H. L., & Chai, K. M. A. (2014). Language modeling with sum-product networks. In *Fifteenth annual conf. of the ISCA*.
Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., et al. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proc. EMNLP, 2014* (pp. 1724–1734).
Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., & Toutanova, K. (2019). BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proc. of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies: Vol. 1 (long and short papers)* (pp. 2924–2936).
Dagan, I., Glickman, O., & Magnini, B. (2005). The PASCAL recognising textual entailment challenge. In *Machine learning challenges workshop* (pp. 177–190). Springer.
Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q., & Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. In *Proc. of the 57th annual meeting of the association for computational linguistics* (pp. 2978–2988).
Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL-HLT* (pp. 4171–4186).
Dolan, W. B., & Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Proc. of the third international workshop on paraphrasing*.
Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.
Grave, E., Joulin, A., & Usunier, N. (2016). Improving neural language models with a continuous cache. ArXiv preprint.
Guo, J., Che, W., Wang, H., & Liu, T. (2014). Learning sense-specific word embeddings by exploiting bilingual resources. In *Proc. of COLING* (pp. 497–507).
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.
Huang, L., Sun, C., Qiu, X., & Huang, X.-J. (2019). GlossBERT: BERT for word sense disambiguation with gloss knowledge. In *Proc. of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing* (pp. 3500–3505).
Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*(1), 193–218.
Inan, H., Khosravi, K., & Socher, R. (2016). Tying word vectors and word classifiers: A loss framework for language modeling. ArXiv preprint.
Jain, S., Bodapati, S. B., Nallapati, R., & Anandkumar, A. (2019). Multi sense embeddings from topic models. arXiv:1909.07746.
Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., & Wu, Y. (2016). Exploring the limits of language modeling. ArXiv preprint.
Khashabi, D., Chaturvedi, S., Roth, M., Upadhyay, S., & Roth, D. (2018). Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proc. of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies: Vol. 1 (long papers)* (pp. 252–262).
Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016). Character-aware neural language models. In: *Proc. AAAI* (pp. 2741–2749).
Kneser, R., & Ney, H. (1995). Improved backing-off for m-gram language modeling. *1*, In *Proc. ICASSP* (pp. 181–184). IEEE.
Kvalseth, T. O. (1987). Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man, and Cybernetics*, *17*(3), 517–519.
Li, J., & Jurafsky, D. (2015). Do multi-sense embeddings improve natural language understanding? In *Proc. EMNLP* (pp. 1722–1732).
Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137.
Loper, E., & Bird, S. (2002). NLTK: the natural language toolkit. ArXiv preprint cs/0205028.
Ma, R., Jin, L., Liu, Q., Chen, L., & Yu, K. (2020). Addressing the polysemy problem in language modeling with attentional multi-sense embeddings. In *ICASSP 2020 - 2020 IEEE international conference on acoustics, speech and signal processing* (pp. 8129–8133). ieeexplore.ieee.org.
Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, *19*(2), 313–330.
Melamud, O., Goldberger, J., & Dagan, I. (2016). context2vec: Learning generic context embedding with bidirectional lstm. In *Proc. of the 20th SIGNLL conf. on computational natural language learning* (pp. 51–61).
Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. ArXiv preprint.
Mikolov, T., Joulin, A., Chopra, S., Mathieu, M., & Ranzato, M. (2014). Learning longer memory in recurrent neural networks. ArXiv preprint.
Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J., & Khudanpur, S. (2011). Extensions of recurrent neural network language model. In *Proc. ICASSP* (pp. 5528–5531). IEEE.
Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proc. NIPS* (pp. 3111–3119).
Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. J. (1990). Introduction to wordnet: An on-line lexical database. *International Journal of Lexicography*, *3*(4), 235–244.
Miller, G. A., Leacock, C., Tengi, R., & Bunker, R. T. (1993). A semantic concordance. In *Proc. of the workshop on human language technology* (pp. 303–308). Association for Computational Linguistics.
Mou, L., Men, R., Li, G., Xu, Y., Zhang, L., Yan, R., et al. (2015). Natural language inference by tree-based convolution and heuristic matching. ArXiv preprint.
Navigli, R., & Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, *193*, 217–250.
Neelakantan, A., Shankar, J., Passos, A., & McCallum, A. (2014). Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proc. EMNLP* (pp. 1059–1069).
Panigrahi, A., Simhadri, H. V., et al. (2019). Word2Sense: sparse interpretable word embeddings. In *Proc. of the 57th*. aclweb.org.
Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., et al. (2018). Deep contextualized word representations. In *Proc. of NAACL-HLT* (pp. 2227–2237).
Pham, N.-Q., Kruszewski, G., & Boleda, G. (2016). Convolutional neural network language models. In *Proc. EMNLP* (pp. 1153–1162).
Pilehvar, M. T., & Camacho-Collados, J. (2019). WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proc. of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies: Vol. 1 (Long and Short Papers)* (pp. 1267–1273).
Qiu, L., Tu, K., & Yu, Y. (2016). Context-dependent sense embedding. In *Proc. EMNLP* (pp. 183–191).
Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.
Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.
Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100, 000+ questions for machine comprehension of text. In *Proc. EMNLP* (pp. 2383–2392).
Reisinger, J., & Mooney, R. J. (2010). Multi-prototype vector-space models of word meaning. In *Human language technologies: The 2010 annual conf. of the North American chapter of the ACL* (pp. 109–117). Association for Computational Linguistics.
Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, *24*(1), 97–123.
Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., et al. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of the 2013 conference on empirical methods in natural language processing* (pp. 1631–1642).
Song, L., Wang, Z., Mi, H., & Gildea, D. (2016). Sense embedding learning for word sense induction. In *Proc. of the fifth joint conf. on lexical and computational semantics* (pp. 85–90).
Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *63*(2), 411–423.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *NIPS* (pp. 5998–6008).
Vial, L., Lecouteux, B., & Schwab, D. (2019). Sense vocabulary compression through the semantic knowledge of wordnet for neural word sense disambiguation. ArXiv preprint arXiv:1905.05677.

Wang, M., Lu, Z., Li, H., Jiang, W., & Liu, Q. (2015). genCNN: A convolutional architecture for word sequence prediction. In *Proc. ACL and the 7th inter. joint conf. on NLP (long papers): Vol. 1* (pp. 1567–1576).

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., et al. (2019). Superglue: A stickier benchmark for general-purpose language understanding systems. ArXiv preprint arXiv:1905.00537.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. ArXiv preprint.

Warstadt, A., Singh, A., & Bowman, S. R. (2019). Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, *7*, 625–641.

White, A. S., Rastogi, P., Duh, K., & Van Durme, B. (2017). Inference is everything: Recasting semantic resources into a unified evaluation framework. In *Proc. of the Eighth international joint conference on natural language processing* (pp. 996–1005).

Williams, A., Nangia, N., & Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *Proc. NAACL-HLT* (pp. 1112–1122).

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, *32*, 5753–5763.

Yuan, D., Richardson, J., Doherty, R., Evans, C., & Altendorf, E. (2016). Semi-supervised word sense disambiguation with neural models. ArXiv preprint.

Zhang, S., Jiang, H., Xu, M., Hou, J., & Dai, L. (2015). The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proc. ACL and the 7th inter. joint conf. on NLP (short papers): Vol. 2* (pp. 495–500).

Zhao, R., & Mao, K. (2017). Topic-aware deep compositional models for sentence classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *25*(2), 248–260.