One-Shot and Few-Shot Exemplification Modeling

John Harvill[1], **Hee Suk Yoon**[2], **Eunseop Yoon**[2], **Mark Hasegawa-Johnson**[1], **Chang D. Yoo**[2]
[1]University of Illinois Urbana-Champaign,
[2]Korea Advanced Institute of Science and Technology
{harvill2, jhasegaw}@illinois.edu, {hskyoon, esyoon97, cd_yoo}@kaist.ac.kr

Abstract

Exemplification modeling is a task where the goal is to produce a viable example sentence that uses a target word with a target definition. The task is non-trivial for polysemous words, and previous works have only explored settings where ample labeled training data is available. In this paper, we demonstrate that exemplification modeling can be performed without a large labeled training corpus by either changing the format of the task (one-shot) or prompting large language models (few-shot), and ablate key components of our proposed one-shot and few-shot systems. We provide extensive automatic and human evaluations of model performance and find that our proposed one-shot and few-shot approaches perform similarly to a fully supervised baseline. We compare and contrast each method in terms of labeled training dataset size, performance, and model size, and find that each technique has at least one tradeoff that another approach does not.

1 Introduction

Many words can have several different meanings depending on the context in which they are used. Given this ambiguity, it is often necessary to resolve the meaning of words in context for the purpose of understanding the semantics of a sentence or learning how to use new words properly. This task is called Word Sense Disambiguation (WSD) and has been widely studied (Navigli, 2009; Scarlini et al., 2020; Barba et al., 2021b). Recently, two related generative tasks have spawned from WSD, namely Definition Modeling (DM) (Bevilacqua et al., 2020) and Exemplification Modeling (EM) (Barba et al., 2021a; He and Yiu, 2022). DM is similar to WSD, except that instead of choosing a word sense from a predefined inventory (classification), a definition is generated for a given word in context. EM can be seen as the inverse task to DM, where an example sentence is generated given a target word and definition. We provide an example input/output pair below:

Input	Output
cool: composure under strain	She kept her <u>cool</u> during the interview.

In this paper, we examine the ability of different systems to perform EM in one-shot or few-shot settings.

One-shot EM. To perform EM in a one-shot fashion (OneEM), we replace the definition with an example sentence using the target word with the intended sense. In this setting, the semantics of the target word are inferred from context in a sentence. We will show that the OneEM format of the task has new applications not possible for EM (see Section 1.1) and can be trained in a self-supervised fashion, requiring only raw text with no word sense labels. An example input/output pair for the OneEM task is provided below:

Input	Output
cool: Drinking a <u>cool</u> beverage is refreshing.	He felt the <u>cool</u> breeze in his hair.

Few-shot EM. We explore the ability of Large Language Models (LLM) to perform EM in a few-shot setting (FewEM). The prompt is created from examples formatted as in Brown et al. (2020), where the FewEM task is performed by completing the last example (See Figure 3).

1.1 Applications

EM can be used for many downstream tasks. Previous works have focused on data augmentation, but we introduce several important applications that further motivate the need for high-quality EM systems and the one-shot format of the task.

Data Augmentation. Previously proposed EM applications are data augmentation for WSD and dictionary example augmentation (He and Yiu, 2022).

Barba et al. (2021a) showed that when used as an augmentation strategy, EM can lead to state-of-the-art performance for WSD.

Vocabulary Learning System. Segler (2007) found that language learners can benefit from exposure to multiple examples using a target word when acquiring new vocabulary. He demonstrates that, in addition to the gloss of a target word, multiple examples can be an integral part of an Intelligent Computer-Aided Language Learning (ICALL) Vocabulary Learning System. A trained OneEM model can serve as an example generator for an ICALL system using as input *any* reading passage, because we can use the context sentence for a new or confusing word as one-shot input for the OneEM model. The benefits of OneEM over EM for this application are: **1)** No sense inventory is needed during training of the OneEM system or at inference time. **2)** When using an EM system for this task, a WSD or DM system would be required first to generate a definition for EM, possibly leading to cascading errors and requiring large amounts of labeled training data. The labeled data constraints imposed by WSD, DM and EM make it difficult to create ICALL systems for low-resourced languages, whereas OneEM does not have these restrictions and can be applied to any language with sufficient raw text for self-supervised training.

Dialog. OneEM (or EM) could also be applied to dialog systems where users express confusion related to vocabulary used in context. Clarifying examples could be generated automatically using the aforementioned ICALL system and integrated into dialog output.

1.2 Contributions

In this paper, we make several contributions: **1)** Introduce one-shot and few-shot versions of Exemplification Modeling (EM). **2)** Propose a self-supervised OneEM system that achieves results similar to a fully-supervised EM system. **3)** Explore the ability of LLMs to perform EM in a few-shot setting (FewEM). **4)** Propose an example verification system that leads to improved performance. **5)** Provide extensive quantitative and qualitative evaluation of generated example sentences.

2 One-Shot Exemplification Modeling

One-shot Exemplification Modeling (OneEM) relies on an example sentence for the semantic signal of the target word sense instead of the gloss. As we

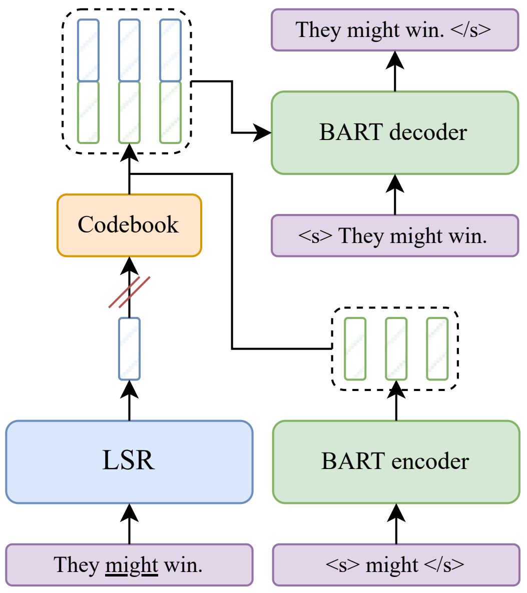


Figure 1: Proposed one-shot training setup. Shown here is the vector quantization disentanglement method that uses a synset codebook, where the red lines indicate cutoff of information leakage from the raw LSR embedding to the decoder. The LSR model and codebook are kept frozen during training. The disentangled LSR is concatenated to all timesteps of the BART encoder output.

will show in the following subsections, OneEM can be trained in a self-supervised fashion, requiring only raw sentences as training data. Self-supervised training is possible due to the empirical observation that bidirectional Language Models (LM) like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) create high-quality representations of words that indicate meaning in context from Masked Language Modeling (MLM) pretraining alone. The one-shot training approach discussed here was proposed previously by Harvill et al. (2023) but is significantly improved in this paper via better disentanglement of information.

One-shot Training. To train a OneEM system in a self-supervised fashion, we use a neural autoencoder and call our approach Sense2Sentence (S2S). Given a sentence s, we want to reconstruct it by conditioning on a target word w and a latent vector representation of the intended target word meaning. We call this vector the Latent Sense Representation (LSR) and denote it as l. We create one-shot training data by selecting w uniformly from s and then extracting l from the LSR module, i.e. $l = \text{LSR}(s, w)$. We then maximize the following probability mass function:

$$p(s|l,w) = \prod_{i=2}^{|s|} p(s_i|s_{1:i-1},l,w) \qquad (1)$$

We model the distribution using a pretrained BART model (Lewis et al., 2020) and train using the cross-entropy loss with teacher forcing. See Figure 1.

Latent Sense Representation. Previous work has shown that word embeddings at the output of a bidirectional transformer-based LM contain information related to the meaning of a given word in context (Vulić et al., 2020b; Liu et al., 2021), so we use these Contextual Word Embeddings (CWE) as the LSR. Concretely, given a target word w, we take the average of all subword embeddings making up w from sentence s.

Word Embedding Extraction. We use a variety of models for the LSR in our experiments and extract CWEs according to best practices for each model type. For the Bi-Encoder Model (BEM) (Blevins and Zettlemoyer, 2020), we use the last layer only since it was trained using that representation. For BERT (Devlin et al., 2019) and a further finetuned model called MirrorWiC (Liu et al., 2021), we average over the last four layers since this was found to be optimal by Liu et al. (2021).

Generating Examples. After training, we can generate new examples in a one-shot setting by using one example sentence to extract the LSR for the given target word sense. Then we sample autoregressively from the decoder using nucleus sampling (top-p) (Holtzman et al., 2019).

3 Disentanglement of Word Meaning

While CWEs contain information relevant to a given word's meaning in context, CWEs also contain information about the sentence in which the word is placed. When using CWEs extracted from pretrained models like BERT (Devlin et al., 2019) as the LSR during training of our proposed OneEM system, we find empirically that there is enough information in the CWEs to make reconstruction of the input sentence easy.[1] This results in the model learning to copy the input sentence and makes it incapable of generating new and diverse examples. Therefore, disentanglement of the target word meaning from other information in the sentence is critical for the proper functioning of our proposed autoencoding system. The two disentanglement methods we explore in this paper are: **1)** finetuning of BERT on WSD using the Bi-Encoder Model (BEM) framework from Blevins and Zettlemoyer (2020) or **2)** Vector quantization via a synset codebook.

3.1 Word Sense Disambiguation Finetuning

The Bi-Encoder Model (BEM) is a WSD system developed by Blevins and Zettlemoyer (2020) where words in context and glosses are encoded separately. Both encoders are pretrained BERT models (Devlin et al., 2019), and the contextual word encoder further refines CWEs to better represent information about a given word's sense. We find empirically when using the BEM contextual word encoder that information about other aspects of the sentence is excluded to such a degree that our autoencoding OneEM training scheme is no longer able to copy its input and the trained system can generate diverse examples. As we will show later in the paper, though, disentanglement is not perfect, because information about input sentence length and the general topic of the input sentence leaks through to the decoder when using this disentanglement method. These effects are seen in the generated sentences by topical overlap and strong correlation between input and generated sentence length.

3.2 Vector Quantization via Synset Codebook

Our second approach towards disentanglement is to use a fixed codebook of vectors. During training and inference, the CWE is extracted and then compared to all codebook vectors via cosine similarity. The most similar code replaces the original CWE and is passed to the BART decoder (see Figure 1). Unlike the WSD disentanglement approach, use of the codebook does not allow information to leak to the decoder and results in a better disentangled representation of target word meaning.

Codebook Construction. To construct the codebook, we extract many CWEs from the LSR model. We then perform K-means clustering on the word embeddings to form concept or synset representations. Due to Zipf's law (Piantadosi, 2014), we know that frequent words occur exponentially more frequently in natural language than infrequent words. For the codebook to represent concepts, we must flatten this distribution when collecting CWEs by clipping the number of occurances of any particular word. For our experiments, we limit the

[1] See Appendix B for further discussion. Figure 5 contains training losses for the various disentanglement techniques discussed in this section.

number of embeddings for any given word to 50 and collect a total of 6M CWEs per LSR model.

3.3 Semi vs. Self-supervision

The main downside to the first disentanglement approach is that BEM requires labeled training data for finetuning on WSD, making the OneEM system that uses this method for disentanglement semi-supervised. The second disentanglement approach is completely self-supervised, and we experiment with BEM (Blevins and Zettlemoyer, 2020), BERT (Devlin et al., 2019) and MirrorWiC (Liu et al., 2021) as the LSR model. Since the autoencoding approach to OneEM is self-supervised, the entire approach is self-supervised when using either BERT (Devlin et al., 2019) or MirrorWiC (Liu et al., 2021) as the LSR model, because neither model uses any labeled data during finetuning.

4 Few-Shot Exemplification Modeling

Given the recent success of LLMs across a variety of tasks (Carlini et al., 2021; Kung et al., 2023; Chen et al., 2021; Austin et al., 2021; Wei et al., 2022; Ouyang et al., 2022), we explore the ability of LLMs to perform EM in a few-shot setting (FewEM). We provide several formatted examples as a prompt to BLOOM or BLOOMZ[2] (Scao et al., 2022) and perform few-shot inference by appending an incomplete example and continuing generation using nucleus sampling (See Figure 3).

4.1 Verification via Definition Modeling

Definition Modeling (DM) (Bevilacqua et al., 2020) can be seen as the inverse of EM, because the definition of the target word is generated based on its use in an example sentence (see Figure 4). We find that, in addition to EM, LLMs are capable of performing DM in a few-shot setting. We capitalize on this ability to verify the quality of generated FewEM examples by passing them as input for few-shot DM and extracting the generated definition (gloss) for the given target word. We then create semantic vector representations of the generated definition and gold gloss using the unsupervised version of SimCSE (Gao et al., 2021). We compute the cosine similarity between gold and generated glosses, sort generated examples of a given word sense in order of decreasing similarity, and choose examples from the beginning of the list for evaluation. The intuition for this process is that if the generated definition of a given target word in a generated FewEM example is similar in meaning to the intended definition, the FewEM example likely uses the target word with the proper meaning.

[2]BLOOMZ is a BLOOM model finetuned on instructions.

5 Baseline

Both previous works on exemplification modeling make use of the ExMaker model (Barba et al., 2021a; He and Yiu, 2022). We reimplement ExMaker from scratch, where we provide the target word (lemma) and definition as input and train the model to maximize the probability of a given labeled example sentence. For direct comparison with our proposed OneEM system, we use the same BART model for the ExMaker baseline. Additionally, we train a sense-agnostic version of the baseline where we exclude the definition and only provide the target word as input. We call this the vanilla version (ExMaker$_V$) and use it to provide a lower bound on EM performance for polysemous words since it cannot take the target definition into account.

6 Data

We focus our evaluation on polysemous words, since EM is a trivial task for monosemous words. We discuss the training, validation and test data for EM, OneEM, and FewEM methods below.

Training. For our proposed OneEM system, we do not require labels and thus train using the 74M raw sentences from BookCorpus (Zhu et al., 2015). For ExMaker, we use Oxford Dictionary[3], which contains 1.4M labeled examples.

Validation. Due to the relatively small size of Oxford Dictionary, we must validate the ExMaker baseline to avoid overfitting. We validate using cross-entropy loss on 142k held-out examples.

Test Set. We create a test set[4] of 167 word senses by hand using polysemous words with two or more distinct meanings (homographs). The goal of manual construction of the dataset is to insure word meanings for the test senses are clearly separate[5] and that gold example sentences are high-quality and easy to read. For each example, we provide the

[3]We use the version prepared by He and Yiu (2022), available at `https://github.com/NLPCode/CDEG`

[4]`https://github.com/jharvill23/OneShotFewShotEM`

[5]For polysemous words, many senses are related and there may be disagreement as to which meanings constitute separate sense entries in a given inventory (see Section 2.1 in Navigli (2009)).

word, lemma, word sense (WordNet 3.0), part-of-speech, definition, and an example sentence. We use the example sentence as input for the OneEM models and as gold data in our evaluations.

7 Experimental Setup

Lemmatization of Target Word. For EM and FewEM, we lemmatize the target word to allow word form flexibility in a given generated sentence. For OneEM, we do not lemmatize due to the main target application for the task, which is a language learning tool that further clarifies the meaning of a word by generating more examples (see Section 1.1). By not lemmatizing the target word for OneEM, we force the generated sentence to produce the target word in the same form as is present in the one-shot example. The main motivation for this is that some words are only homographs in certain forms. For example, the word "saw" is a homograph, because it can refer to the past tense of the verb "to see" and the noun meaning "instrument used to cut wood." If we were to lemmatize ("saw"$\rightarrow$"see"), we could produce an incorrect example depending on which meaning was used in the one-shot input example.
Training and Validation Hyperparameters. We train for 500k steps and set batch size to 64 for ExMaker and Sense2Sentence (S2S) models. We validate ExMaker every 25k steps and use the 500k checkpoint for S2S.
Generation. We use $p = 0.5$ for nucleus sampling for all methods (ExMaker, S2S, LLM). For definition modeling (DM) verification, we use three beams during beam search due to GPU memory constraints (see Appendix C).

7.1 Additional One-Shot Configurations

Contextual Representation. In addition to using BERT for the LSR, we also experiment with a self-supervised method for refining contextual word embeddings called MirrorWiC (Liu et al., 2021).
Codebook Size. To examine the effect of the codebook size on performance, we run experiments using codebooks of size 1k, 20k, 100k, and 150k for the MirrorWiC LSR model.

8 Evaluations

We want to evaluate three aspects of generated example sentences: **1)** How well the target word takes on its intended meaning (semantic match), **2)** Fluency of text, **3)** Diversity of generated examples from one another. We measure diversity automatically and use both automatic and human evaluations to measure semantic match and fluency. For all OneEM approaches, we compute Pearson's correlation coefficient between input and output sentence lengths to provide insight into how well each LSR disentanglement method can remove information unrelated to target word meaning.

8.1 Diversity

Self-BLEU. For a given method, we use Self-BLEU (Zhu et al., 2018) to measure diversity of generated sentences for any given word sense by using one example as the hypothesis and computing the BLEU score with respect to the remaining generated sentences (references). Self-BLEU scores are computed using five generated sentences per word sense and averaged over all test word senses, where a lower score indicates better diversity due to less n-gram overlap with other sentences.
Vector Semantic Distance. Current sentence semantic encoders create semantic vector representations that correlate well with human judgments (Gao et al., 2021; Chuang et al., 2022). Given this ability, we also measure diversity using the distance between two sentences in the semantic vector space and use SimCSE (Gao et al., 2021) as the encoder. For a given word sense, we use five generated sentences and compute the cosine distance between each pair. We take the average over all word senses and call this the Vector Semantic Distance (VSD). Higher VSD indicates more semantic diversity among generated sentences.

8.2 Coherence

We use "coherence" here similarly to Barba et al. (2021a) to describe the quality of a given generated sentence. Both semantic match and fluency are necessary for a generated example to be considered coherent for EM, and we evaluate these two aspects automatically and using human annotators. We provide brief descriptions of human evaluations for semantic match and fluency in this section, so refer to Appendix D for further details about annotators and evaluation.
Semantic Match. For the automatic evaluation, we extract the Contextual Word Embedding (CWE) of the target word from the generated sentence and compare to its ARES sense embedding (Scarlini et al., 2020) via cosine similarity as is done by Barba et al. (2021a). Higher similarity indicates a closer match of the target word to its intended

Model	Task	LSR Model	Vector Quant.	Train. Data	# Param. (billion)	Verif.	In/Out $\rho\downarrow$	Diversity SB4↓	Diversity VSD↑	Coherence SM↑	Coherence F↑
Self-supervised											
$\mathrm{S2S}_{150k}$	OneEM	MWiC	Yes	BC	0.25	No	0.10	0.52	0.58	0.33	0.86
$\mathrm{S2S}_{20k}$	OneEM	MWiC	Yes	BC	0.25	No	0.07	0.52	0.58	0.32	0.92
$\mathrm{S2S}_{1k}$	OneEM	MWiC	Yes	BC	0.25	No	0.06	0.53	0.59	0.30	0.96
$\mathrm{S2S}_{100k}$	OneEM	MWiC	Yes	BC	0.25	No	0.20	0.53	0.56	0.33	0.86
$\mathrm{S2S}_{100k}$	OneEM	BERT	Yes	BC	0.25	No	0.06	0.51	0.63	0.32	0.94
S2S	OneEM	BERT	No	BC	0.25	No	0.90	0.77	0.27	0.36	0.95
$\mathrm{ExMaker}_{\mathrm{v}}$	EM	n.a.	n.a.	OD	0.14	No	n.a.	0.46	0.80	0.21	0.80
Semi-supervised											
$\mathrm{S2S}_{100k}$	OneEM	BEM	Yes	BC	0.25	No	0.04	0.53	0.60	0.33	0.83
S2S	OneEM	BEM	No	BC	0.25	No	0.85	0.71	0.35	0.36	0.95
Fully-supervised											
ExMaker	EM	n.a.	n.a.	OD	0.14	No	n.a.	0.51	0.61	0.36	0.76
Few-shot (LLM)											
BLOOMZ_{7B}	FewEM	n.a.	n.a.	n.a.	7.1	Yes	n.a.	0.62	0.46	0.33	0.97
BLOOMZ_{7B}	FewEM	n.a.	n.a.	n.a.	7.1	No	n.a.	0.59	0.51	0.31	0.94
BLOOMZ_{1B}	FewEM	n.a.	n.a.	n.a.	1.1	Yes	n.a.	0.57	0.51	0.31	0.95
BLOOMZ_{1B}	FewEM	n.a.	n.a.	n.a.	1.1	No	n.a.	0.56	0.55	0.29	0.93
BLOOM_{7B}	FewEM	n.a.	n.a.	n.a.	7.1	Yes	n.a.	0.62	0.46	0.34	0.98
BLOOM_{7B}	FewEM	n.a.	n.a.	n.a.	7.1	No	n.a.	0.58	0.54	0.32	0.97
Reference											
Gold	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	0.37	0.97

Table 1: Automatic evaluations. Abbreviations are as follows - MWiC: MirrorWiC, BC: BookCorpus, OD: Oxford Dictionary, SB4: Self-BLEU (4-gram), VSD: Vector Semantic Distance, SM: Semantic Match, F: Fluency. The "Verif." column refers to whether Definition Modeling verification (LLM) was used. In/Out ρ is the Pearson Correlation Coefficient between input and output sentence length.

Model	Task	LSR Model	Vector Quant.	Training Data	# Param. (billion)	Verif.	Coherence $\mathrm{SM}_H\uparrow$	Coherence $\mathrm{F}_H\uparrow$
Self-supervised								
$\mathrm{S2S}_{150k}$	OneEM	MWiC	Yes	BC	0.25	No	4.35	4.89
Semi-supervised								
S2S	OneEM	BEM	No	BC	0.25	No	4.82	4.78
Fully-supervised								
ExMaker	EM	n.a.	n.a.	OD	0.14	No	4.70	4.83
Few-shot (LLM)								
BLOOM_{7B}	FewEM	n.a.	n.a.	n.a.	7.1	Yes	4.78	4.96
Reference								
Gold	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	4.98	4.99

Table 2: Human evaluations. Abbreviations are the same from Table 1.

meaning. For human evaluations, we ask annotators to rate example sentences on a scale from zero to five, where five indicates the target word exactly matches its intended meaning.

Fluency. For our automatic evaluation, we use a classifier[6] finetuned on the CoLA task (Warstadt et al., 2019), which judges the grammatical acceptability of a given sentence. We report the average probability of acceptability of generated sentences for each method. For human evaluations, we ask annotators to provide a score for each sentence from zero to five, where five means the sentence is perfectly grammatical.

[6]https://huggingface.co/textattack/roberta-base-CoLA

9 Results

Numerical results for automatic and human evaluations are given in Tables 1 and 2, respectively. We discuss several key takeaways below.

Self-Supervised Performance (OneEM). Compared to gold examples, those generated using the self-supervised approach are of high quality based on both automatic and human evaluations. The fully-supervised EM baseline, ExMaker, only slightly outperforms the best self-supervised method (row 1 in Tables 1 and 2) for semantic match and fluency, and achieves almost identical diversity values.

We find that codebook size only has a small effect on performance, where we notice a slight

Word 1: fan **POS**: noun	**Definition**: An ardent follower and admirer **One-shot example**: I've been a fan of Hemingway's writing for years.
S2S$_{150k}$ (MWiC)	I wasn't a big fan of the slow motion movies, but I had to admit, I liked the latter.
S2S (BEM)	But she is a fan of the book of Juliet, and she likes to read.
ExMaker	He is a fan of the arts, and a huge supporter of the arts in general.
BLOOM$_{7B}$	The Beatles were a huge fan of Elvis Presley.
Word 2: fan **POS**: noun	**Definition**: A device for creating a current of air by movement of a surface or surfaces **One-shot example**: She used a large fan during the summer to keep her shop cool.
S2S$_{150k}$ (MWiC)	The voice in the room seemed to be some kind of a gas-powered fan.
S2S (BEM)	The house was fitted with a fan and made the people to love the air.
ExMaker	A pilot fan is used to provide air for the exhaust system.
BLOOM$_{7B}$	A fan is a device that creates a current of air by moving a surface or surfaces.
Word 3: lie **POS**: verb	**Definition**: Assume a reclining position **One-shot example**: I'm tired and need to go lie down.
S2S$_{150k}$ (MWiC)	No, you don't have to lie down, said Joshua.
S2S (BEM)	I just need to go and stretch out to lie down.
ExMaker	We were both watching the kids lie down on the couch.
BLOOM$_{7B}$	The boy lay down on the bed and fell asleep.
Word 4: lie **POS**: verb	**Definition**: Tell an untruth; pretend with intent to deceive **One-shot example**: I need to know what's really going on, so don't lie to me.
S2S$_{150k}$ (MWiC)	She had a feeling that he wouldn't lie to her.
S2S (BEM)	I need to be able to hear you, so do it, then try to lie to me.
ExMaker	He was willing to lie and deceive, but was not prepared to give his life for the sins of the Devil.
BLOOM$_{7B}$	The boy lies that he has a friend in the next room.
Word 5: object **POS**: noun	**Definition**: A tangible and visible entity; an entity that can cast a shadow **One-shot example**: She picked up the mysterious object and looked closely at it.
S2S$_{150k}$ (MWiC)	She put the object back in its sheath and stood up.
S2S (BEM)	He looked at the mysterious object and reached into his pocket for it.
ExMaker	The ship, in fact, was a floating object, casting a shadow over the whole area.
BLOOM$_{7B}$	The house was built of solid stone, and the object was a tower.
Word 6: object **POS**: verb	**Definition**: Express or raise an objection or protest or criticism or express dissent **One-shot example**: He didn't object to the new proposal, but he wished the timeline was a little more spaced out.
S2S$_{150k}$ (MWiC)	The captain was a bit reluctant to object.
S2S (BEM)	She didn't object to the idea of the new program, but she wanted to get her head straight.
ExMaker	It is wrong to object to a positive review of a health service for any reason.
BLOOM$_{7B}$	I object to the fact that the president is trying to take away the right to bear arms.

Table 3: Generation examples for different word senses.

reduction in automatic semantic match score as the codebook size shrinks. Additionally, we find a slight improvement in semantic match when using the refined CWEs extracted from MirrorWiC (Liu et al., 2021) compared to those extracted from BERT (Devlin et al., 2019), indicating that CWEs from MirrorWiC produce higher quality representations of semantics in context.

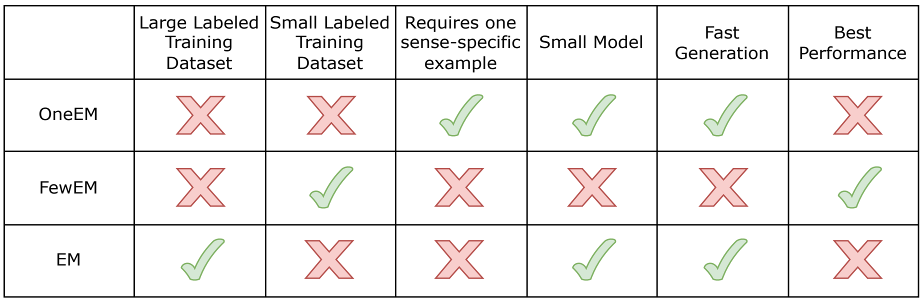
Disentanglement (OneEM). Based on the input/output length correlation shown in Table 1, we can see that Vector Quantization (VQ) significantly outperforms WSD finetuning in terms of disentanglement. For OneEM methods without VQ, we see a correlation coefficient close to one, whereas OneEM methods using VQ achieve correlation coefficients close to zero. This indicates that information related to the length of the one-shot example sentence is almost entirely removed when using VQ. Improved disentanglement also leads to improved diversity scores, where we see that VQ methods achieve Self-BLEU scores close to 0.5 whereas approaches without VQ achieve scores in the range 0.7-0.8. Finally, by comparing diversity and input/output correlation scores for BEM and BERT with no VQ in Table 1, we see that WSD finetuning improves performance compared to using no disentanglement approach, but only marginally.

LLM Performance (FewEM). Overall, LLMs are able to perform Few-shot Exemplification Modeling (FewEM) extremely well, effectively matching or outperforming all other methods on human evaluations, while achieving slightly worse diversity scores. We also find performance improvements when using Definition Modeling (DM) verification to filter bad examples. There does not appear to be much difference between models with and without instruction finetuning (Wei et al., 2021) (BLOOMZ vs. BLOOM), but we see the best performance from BLOOM.

10 Generation Examples

We provide some generated examples drawn randomly from our human evaluations in Table 3. We show two noun senses for the word "fan," two verb senses for "lie," and a noun and verb sense for "object." Overall, the examples are fluent and use the target word with the proper meaning, but there are a few weaknesses that we highlight below.

Logic. In some cases an example sentence is grammatical and makes it clear which sense is being



	Large Labeled Training Dataset	Small Labeled Training Dataset	Requires one sense-specific example	Small Model	Fast Generation	Best Performance
OneEM	✗	✗	✓	✓	✓	✗
FewEM	✗	✓	✗	✗	✗	✓
EM	✓	✗	✗	✓	✓	✗

Figure 2: Pros and Cons of EM, OneEM and FewEM.

used for the target word, but overall the sentence does not make sense. For example, the sentence generated by S2S$_{150k}$ (MWiC) for word 2 obviously uses the word "fan" properly (gas-powered fan), but says a voice is a gas-powered fan, which is illogical.

Trivial Example. For those methods performing traditional Exemplification Modeling (EM), where the target word and definition are provided as input, the models occasionally create a trivial example by stating the definition of the target word. For example, the BLOOM$_{7B}$ example for word 2 simply defines the word "fan" using the provided definition. We see traces of the definition show up for ExMaker as well in the example for word 5; "cast a shadow" is in the definition, and "casting a shadow" is in the generated example. Such behavior is somewhat expected given that autoregressive generation is prone to producing repetitions (Holtzman et al., 2019), but future work may mitigate this issue by using a repetition penalty during decoding (Keskar et al., 2019) or filtering out examples with high n-gram overlap with respect to the gloss.

11 Discussion

The approaches to EM, OneEM, and FewEM discussed in this paper each have benefits and drawbacks, which are summarized in Figure 2. Overall, we find there is a tradeoff between amount of labeled training data and model size needed for good performance. Based on human evaluations, we see the best performance from the FewEM setting, which uses a Large Language Model (LLM). Similar performance can be achieved using a much smaller model (ExMaker), but requires a large amount of labeled training data. Finally, we can achieve similar performance to ExMaker when using a small model without a large corpus of labeled training data (self-supervised training), but require at least one example at inference time in order to generate more examples (OneEM).

12 Conclusions

In this paper, we proposed two variations of the Exemplification Modeling (EM) task, namely One-shot EM (OneEM) and Few-shot EM (FewEM). We discussed novel applications and described a self-supervised solution for the OneEM task, and ablated several configurations (codebook size, CWE model) to better understand performance. We also performed extensive experiments using LLMs to solve the FewEM task, and found that a pretrained LLM can perform FewEM extremely well when prompted with only a few examples. We additionally studied the use of few-shot Definition Modeling (DM) to verify FewEM examples and found that such an approach leads to improved performance. In addition to being useful for downstream applications like data augmentation, ICALL vocabulary learning systems, and dialog systems, the family of EM tasks discussed in this paper provide evidence that pretrained language models have a strong understanding of differences between various word senses. Our experiments demonstrate that target word meaning can be inferred from either a definition (EM, FewEM) or an in-context example (OneEM), providing flexibility for example generation depending on downstream task constraints.

13 Limitations

Each approach discussed in this paper has at least one undesirable limitation. The baseline, ExMaker (Barba et al., 2021a), works well but requires a large, labeled dataset for training. Our proposed one-shot system trains in a self-supervised fashion, but requires at least one example at inference time to be able to generate more examples. The few-shot system requires only a handful of annotated examples, but relies on a Large Language Model

(LLM), which is costly to train. A combination of these systems may be better suited to different downstream applications, but we leave exploration of this kind to future work.

14 Ethics Statement

We rely on pretrained models for our experiments, and biases present in training data may surface when using our proposed systems (Nadeem et al., 2021; Liang et al., 2021). We do not actively focus our efforts in this paper towards controlling for such biases, so it is important to exercise caution when using generated example sentences in downstream applications.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Edoardo Barba, Luigi Procopio, Caterina Lacerra, Tommaso Pasini, and Roberto Navigli. 2021a. Exemplification modeling: Can you give me an example, please? In *IJCAI*, pages 3779–3785.

Edoardo Barba, Luigi Procopio, and Roberto Navigli. 2021b. Consec: Word sense disambiguation as continuous sense comprehension. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1492–1503.

Michele Bevilacqua, Marco Maru, and Roberto Navigli. 2020. Generationary or "how we went beyond word sense inventories and learned to gloss". In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7207–7221, Online. Association for Computational Linguistics.

Terra Blevins and Luke Zettlemoyer. 2020. Moving down the long tail of word sense disambiguation with gloss informed bi-encoders. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1006–1017, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *USENIX Security Symposium*, volume 6.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljacic, Shang-Wen Li, Scott Yih, Yoon Kim, and James Glass. 2022. DiffCSE: Difference-based contrastive learning for sentence embeddings. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4207–4218, Seattle, United States. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

John Harvill, Mark Hasegawa-Johnson, Hee Suk Yoon, Chang D. Yoo, and Eunseop Yoon. 2023. One-shot exemplification modeling via latent sense representations. In *Proceedings of the 8th Workshop on Representation Learning for NLP (RepL4NLP 2023)*, pages 303–314, Toronto, Canada. Association for Computational Linguistics.

Xingwei He and Siu Ming Yiu. 2022. Controllable dictionary example generation: Generating example sentences for specific targeted audiences. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 610–627.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv e-prints*, pages arXiv–1904.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Tiffany H Kung, Morgan Cheatham, Arielle Medenilla, Czarina Sillos, Lorie De Leon, Camille Elepaño, Maria Madriaga, Rimel Aggabao, Giezel Diaz-Candido, James Maningo, et al. 2023. Performance of chatgpt on usmle: Potential for ai-assisted medical education using large language models. *PLoS digital health*, 2(2):e0000198.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2021. Towards understanding and mitigating social biases in language models. In *International Conference on Machine Learning*, pages 6565–6576. PMLR.

Qianchu Liu, Fangyu Liu, Nigel Collier, Anna Korhonen, and Ivan Vulić. 2021. MirrorWiC: On eliciting word-in-context representations from pretrained language models. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 562–574, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online. Association for Computational Linguistics.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):1–69.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Steven T Piantadosi. 2014. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic bulletin & review*, 21:1112–1130.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020. 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 (EMNLP)*, pages 3528–3539.

Thomas M Segler. 2007. Investigating the selection of example sentences for unknown target words in icall reading texts for l2 german.

Ivan Vulić, Simon Baker, Edoardo Maria Ponti, Ulla Petti, Ira Leviant, Kelly Wing, Olga Majewska, Eden Bar, Matt Malone, Thierry Poibeau, Roi Reichart, and Anna Korhonen. 2020a. Multi-SimLex: A large-scale evaluation of multilingual and crosslingual lexical semantic similarity. *Computational Linguistics*, 46(4):847–897.

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. 2020b. Probing pretrained language models for lexical semantics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7222–7240, Online. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

Few-Shot Exemplification Modeling

Word 1: tie
Part of speech 1: noun
Definition 1: a result in a game or other competitive situation in which two or more competitors or teams have the same score or ranking
Example 1: The game ended in a tie even though the Rangers seemed to have outplayed the Devils.

Word 2: address
Part of speech 2: verb
Definition 2: to give a speech to a group of people
Example 2: The principal went up to the podium in order to address the graduating student body.

.
.
.

Word 11: match
Part of speech 11: noun
Definition 11: a flammable material that can be ignited by friction
Example 11: He struck a match and lit a candle.

Figure 3: Few-Shot Exemplification Modeling. Task is inferred by placing the target word and definition first, followed by the example sentence. The test instance is given last with the example sentence field empty, and the model generates text using nucleus sampling. Prompt text is black and generated text is blue.

Few-Shot Definition Modeling

Word 1: tie
Example 1: The game ended in a tie even though the Rangers seemed to have outplayed the Devils.
Definition 1: a result in a game or other competitive situation in which two or more competitors or teams have the same score or ranking

Word 2: address
Example 2: The principal went up to the podium in order to address the graduating student body.
Definition 2: to give a speech to a group of people

.
.
.

Word 11: match
Example 11: He struck a match and lit a candle
Definition 11: a small piece of wood that can be used to start a fire

Figure 4: Few-Shot Definition Modeling. Task is inferred by placing the target word and example sentence first, followed by the definition. The test instance is given last with the definition field empty, and the model generates text using beam search.

A Few-shot Exemplification and Definition Modeling

We provide visual examples of the input format for Few-shot Exemplification Modeling and Definition Modeling in Figures 3 and 4, respectively.

B Disentanglement

Training losses for various one-shot settings are shown in Figure 5. Note that the loss is much smaller when using BERT embeddings for the LSR, indicating that a lot of information is able to leak from the input sentence to the decoder. When using vector quantization, much less information is leaked and disentanglement of target word meaning is better.

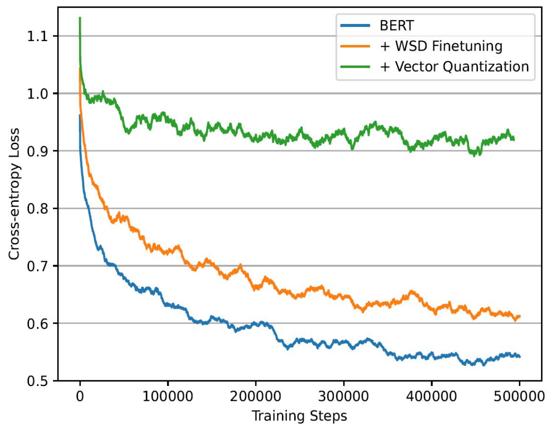


Figure 5: Training losses for one-shot approach when using different embeddings for LSR. We show vanilla BERT embeddings (BERT), BERT embeddings after WSD finetuning (+ WSD Finetuning) and vector-quantized (codebook) BERT embeddings (+ Vector Quantization).

C Computational Details

We use two NVIDIA RTX 3090 Ti GPUs to run our experiments. For the one-shot approach, contextual word embedding extraction for codebook construction takes approximately four days. The K-means clustering step takes approximately eight hours.[7] Model training for the largest codebook (150k) takes approximately five days.[8]

D Human Evaluations

The full set of instructions sent to the annotators is available on GitHub.[9] We discuss other relevant details below.

Recruitment. We recruit three volunteer annotators that are acquaintances of the authors, where all annotators live in the United States. We informed the annotators that all annotations would be kept private and would only be used for evaluation of our models.

Dataset Construction. We use a subset of 80 word senses per method from our test dataset to make the annotation workload manageable for our volunteer annotators and evaluate one example per word sense.

Inter-Annotator Agreement. Since judgments are inherently subjective, we use the Average Mean Inter-Annotator Agreement (AMIAA) (Vulić et al., 2020a), which measures how well the *ranks* of the samples from each annotator match. This measurement thus requires that all samples are evaluated by each annotator, which is why we forego using an online annotation tool such as Amazon Mechanical Turk and must rely on volunteers willing to annotate a large number of samples. The formula for AMIAA is given below in Equation 2:

$$\text{AMIAA} = \frac{1}{K}\sum_i \rho(s_i, \mu_i),$$
$$\text{where } \mu_{i,n} = \frac{1}{K-1}\sum_{j\neq i} s_{j,n} \quad (2)$$

where K refers to the number of annotators, s_i refers to the scores for annotator i, and μ_i refers to the average scores when leaving out annotator i. For our experiments, we find AMIAA values of 0.39 for both fluency and semantic match, respectively, indicating moderate agreement.

[7]We accelerate K-means computation on one GPU with FAISS (https://faiss.ai/).

[8]Training without a codebook takes approximately 1.5 days, so the extra training time comes from the codebook lookup step for vector quantization.

[9]https://github.com/jharvill23/OneShotFewShotEM