Université de Montréal

Autoencoders for natural language semantics

par

Tom Bosc

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en Informatique

September 6, 2022

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Autoencoders for natural language semantics

présentée par

Tom Bosc

a été évaluée par un jury composé des personnes suivantes :

Aaron Courville
(président-rapporteur)

Pascal Vincent
(directeur de recherche)

Aishwarya Agrawal
(membre du jury)

Timothy O'Donnell
(examinateur externe)

Alexandru Hanganu
(représentant du doyen de la FESP)

Résumé

Les auto-encodeurs sont des réseaux de neurones artificiels qui apprennent des représentations. Dans un auto-encodeur, l'encodeur transforme une entrée en une représentation, et le décodeur essaie de prédire l'entrée à partir de la représentation. Cette thèse compile trois applications de ces modèles au traitement automatique des langues : pour l'apprentissage de représentations de mots et de phrases, ainsi que pour mieux comprendre la compositionnalité.

Dans le premier article, nous montrons que nous pouvons auto-encoder des définitions de dictionnaire et ainsi apprendre des *vecteurs de définition*. Nous proposons une nouvelle pénalité qui nous permet d'utiliser ces vecteurs comme entrées à l'encodeur lui-même, mais aussi de les mélanger des vecteurs distributionnels pré-entraînés. Ces vecteurs de définition capturent mieux la similarité sémantique que les méthodes distributionnelles telles que word2vec. De plus, l'encodeur généralise à un certain degré à des définitions qu'il n'a pas vues pendant l'entraînement.

Dans le deuxième article, nous analysons les représentations apprises par les auto-encodeurs variationnels séquence-à-séquence. Nous constatons que les encodeurs ont tendance à mémoriser les premiers mots et la longueur de la phrase d'entrée. Cela limite considérablement leur utilité en tant que modèles génératifs contrôlables. Nous analysons aussi des variantes architecturales plus simples qui ne tiennent pas compte de l'ordre des mots, ainsi que des méthodes basées sur le pré-entraînement. Les représentations qu'elles apprennent ont tendance à encoder plus nettement des caractéristiques globales telles que le sujet et le sentiment, et cela se voit dans les reconstructions qu'ils produisent.

Dans le troisième article, nous utilisons des simulations d'émergence du langage pour étudier la compositionnalité. Un locuteur – l'encodeur – observe une entrée et produit un message. Un auditeur – le décodeur – tente de reconstituer ce dont le locuteur a parlé dans son message. Nous émettons l'hypothèse que faire des phrases impliquant plusieurs entités, telles que « Jean aime Marie », nécessite fondamentalement de percevoir chaque entité comme un tout. Nous dotons certains agents de cette capacité grâce à un mechanisme d'attention, alors que d'autres en sont privés. Nous proposons différentes métriques qui mesurent à quel point les langues des agents sont naturelles en termes de structure d'argument, et si elles sont

davantage analytiques ou synthétiques. Les agents percevant les entités comme des touts échangent des messages plus naturels que les autres agents.

Mots-clés: Traitement Automatique des Langues, Apprentissage Automatique, Intelligence Artificielle, Sémantique, Autoencodeur, Apprentissage Profond, Réseaux de Neurones Artificiels

Abstract

Autoencoders are artificial neural networks that learn representations. In an autoencoder, the encoder transforms an input into a representation, and the decoder tries to recover the input from the representation. This thesis compiles three different applications of these models to natural language processing: for learning word and sentence representations, as well as to better understand compositionality.

In the first paper, we show that we can autoencode dictionary definitions to learn word vectors, called *definition embeddings*. We propose a new penalty that allows us to use these definition embeddings as inputs to the encoder itself, but also to blend them with pretrained distributional vectors. The definition embeddings capture semantic similarity better than distributional methods such as word2vec. Moreover, the encoder somewhat generalizes to definitions unseen during training.

In the second paper, we analyze the representations learned by sequence-to-sequence variational autoencoders. We find that the encoders tend to memorize the first few words and the length of the input sentence. This limits drastically their usefulness as controllable generative models. We also analyze simpler architectural variants that are agnostic to word order, as well as pretraining-based methods. The representations that they learn tend to encode global features such as topic and sentiment more markedly, and this shows in the reconstructions they produce.

In the third paper, we use language emergence simulations to study compositionality. A speaker – the encoder – observes an input and produces a message about it. A listener – the decoder – tries to reconstruct what the speaker talked about in its message. We hypothesize that producing sentences involving several entities, such as "John loves Mary", fundamentally requires to perceive each entity, John and Mary, as distinct wholes. We endow some agents with this ability via an attention mechanism, and deprive others of it. We propose various metrics to measure whether the languages are natural in terms of their argument structure, and whether the languages are more analytic or synthetic. Agents perceiving entities as distinct wholes exchange more natural messages than other agents.

Keywords: Natural Language Processing, Machine Learning, Artificial Intelligence, Semantics, Autoencoder, Deep Learning, Artificial Neural Networks

Contents

List of Tables

the ability to deal with negation. The baseline models also fail to capture the sentiment on the last example, and copy the beginning on the first three examples. 156

List of Figures

Liste des sigles et des abréviations

ERM	Empirical Risk Minimization
ELBo	Evidence Lower Bound
KL	Kullback-Leibler divergence
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NLL	Negative Log Likelihood
NLP	Natural Language Processing
RNN	Recurrent Neural Network
VAE	Variational AutoEncoder

Acknowledgement

First of all, I would like to thank my supervisor Pascal Vincent. I am incredibly lucky and honored that Pascal accepted me as a student. He shared his invaluable experience and time with me, as well as his valuable financial support. Pascal knew when to give me freedom to explore, while at the same time keeping me on track. Besides his skills, Pascal is kind and enthusiastic which made our collaborations very enjoyable, and the importance of the human factor cannot be overstated. Thank you for the trip, Pascal!

I also thank my collaborators Dzmitry Bahdanau, Stanisław Jastrzębski, Edward Grefenstette, Yoshua Bengio, Maksym Korablyov, Andrei Nica, Jarrid Rector-Brooks, Kanika Madan, Nikolay Malkin, Emmanuel Bengio. Special recognition goes to Dzmitry for inviting me to collaborate early on and helping me find my footing. I am also especially grateful to Maksym for sharing his contagious passion for biology and introducing me to GFlowNets.

I thank Michael Auli and Marc'Aurelio Ranzato who kindly accepted to supervise me during my internship at Facebook in Menlo Park, as well as Maria Doliashvili, Jean Maillard and Sidak Pal Singh who made my life in California very enjoyable despite my broken wrist.

I also thank all my Montreal friends. They have all enriched my life in a specific way. This includes, but is not limited to Adam Aqariden, Rim Assouel, Laura Ball, Dishank Bansal, Hugo Bérard, Alexandre de Brébisson, Paul-José Brunetti, Gauthier Gidel, Gabriel Huang, Grace Huang, Chang-Hun, David Kanaa, Arnaud Lafortune, Salem Lahlou, Rémi Le Priol, Elise Michely, Evgenii Nikishin, Mélisande Teng, Christos Tsirigotis, Ahmed Touati, Victor Schmidt. I miss you all already!

To my cherished friends from France, including Kévin de Benedetti, Vadim Bertrand, Thibault Bram, Damien Dénéréaz, Emile Dumas, Théodore Ineich, Sophie Lély, Flavien Maggengo, Etienne Pradere, François Rebatel, Thibault Rihet, Raphaël Vaton - I send all my love and gratitude for being loyal after so many years.

Finally, I thank my parents Isabelle and Etienne, my paternal grand-parents Simone and Fernand, as well as my brothers Félix and Arthur for their support and affection.

Chapter 1

Introduction

The field of *natural language processing* (NLP) is concerned with programs and machines that understand and produce natural language. The NLP community tries to automate various tasks such as answering questions, translating documents from one language to another, summarizing a large set of documents or paraphrasing, etc. NLP is usually restricted to written language, leaving aside speech recognition and speech synthesis.

Nowadays, NLP tasks are mostly framed as *machine learning* (ML) tasks, in particular, as *supervised learning* tasks. The behavior of the desired program is determined by a dataset containing inputs and outputs. ML algorithms produce models that predicts the outputs given the inputs. For example, a model trained to perform machine translation observes as an input a sentence in language A and tries to predict as an output a translation in language B [Brown et al., 1990]. Among ML methods, deep learning [LeCun et al., 2015, Schmidhuber, 2015] have come to dominate. Deep learning models are artificial neural networks, functions of millions or billions of parameters. During *learning*, these parameters are iteratively modified to reduce the prediction error, i.e. the discrepancy between the ground-truth output given in the dataset, and the prediction of the model.

Unfortunately, learning is inherently difficult. Supervised learning with parametric models amounts to inductive inference [Shalev-Shwartz and Ben-David, 2014]: we want to find a model that is consistent with our data and predicts the correct outputs, but there are usually infinitely many such predictors. In fact, the optimal predictor might not even be considered by our search procedure. Furthermore, there is generally not enough data to reach optimal performance. Labelled data is relatively scarce since the outputs that we want to predict usually require human labor, and are thus expensive to produce.

Unsupervised learning is an attractive solution to the lack of labelled data. This form of learning does not require outputs to be labelled. To understand how this works, take the example of word representations. In deep learning, it is common to represent words as continuous vectors. These vectors are optimised *end-to-end* to minimize the prediction error,

just like any other parameter of the model [Bengio et al., 2003]. Indeed, the ability of deep neural networks to extract features is one of their biggest strengths [Bengio et al., 2013]. However, due to the general fact that datasets are too small, most words will be missing from our training data, and simply not represented (*out-of-vocabulary*). Thus, the algorithms will underperform when exposed to new words. Worse, words that appear only a handful of times or only in a single sense can create spurious correlations. To deal with these issues, we can learn generic word vectors that can be used with any downstream task [Schütze, 1992, Mikolov et al., 2013a]. These vectors are *distributional*, in the sense that the representation of a word w reflects the distribution of words in its surroundings [Harris, 1954]. They are trained efficiently, on corpora that are so large that the rare words of our labelled data appear frequently there. As a result, they can then be used as inputs to solve downstream tasks with great accuracy [Collobert and Weston, 2008, Turian et al., 2010].

Autoencoders are a popular class of unsupervised learning algorithms [Rumelhart et al., 1986, Hinton et al., 1995, Vincent et al., 2008, Rifai et al., 2011, Kingma and Welling, 2013, Devlin et al., 2018]. An autoencoder is made of two smaller models: an encoder, which transforms a datapoint x into some vector h, and a decoder, which tries to invert this mapping and reconstruct x from h. There is a variety of different autoencoders. Some can be used for learning representations, others for generating new data, some of them for both purposes.

In this thesis, we use autoencoders for various purposes that are related to *semantics*. Semantics is used in a broad sense here as being related to questions of meaning, in contrast with grammar. Thus, we are not only, or not directly, concerned by NLP tasks such as morphological analysis (determining the components making up words), part-of-speech tagging (determine whether a word is a noun, verb, adjective, etc.), syntactic parsing (computing syntactic trees of sentences), etc. [Jurafsky and Martin, 2009].

In Chapter 5, we present an autoencoder that reads dictionary definitions x and produces word vectors h. This encoder has interesting properties compared to distributional methods. First, it is data efficient. A single definition is enough to compute a representation, in contrast to distributional methods which require seeing a word in different contexts. Second, the information captured by the vectors has a slightly different nature than what distributional vectors encode. Arguably, the definitions that are used as inputs are themselves representations of the things that are denoted by the words, not representations of these words. Hence we see that the vectors produced are better at capturing the similarity relation – the relation that holds between "tea" and "coffee", which are both beverages – rather than the association relation – for example, between "tea" and "cup", words which frequently co-occur. Thirdly, we introduce a penalty that encourages the representations produced by the encoder to be reused as inputs. This can be leveraged to simply improve performance, or to incorporate information from pretrained distributional word embeddings. Moreover, it could potentially enable the network to self-improve by using the weights it predicts.

In Chapter 7, we show how to improve controllable text generation with *sequence-to-sequence variational autoencoders* (seq2seq-VAEs) [Kingma and Welling, 2013, Bowman et al., 2016]. These VAEs use *Long Short-Term Memories* [Hochreiter and Schmidhuber, 1997] (LSTMs) as encoders and decoders. They are notoriously hard to train, and many researchers have blamed optimisation. However, the loss function is underspecified: there are many different solutions to the optimisation problem. We propose a simple method to look at what information is stored in its hidden representation h. We find that it mostly encodes information about the first few words and the number of words in the text. As a consequence, it is hard or impossible to use this model to produce texts with a global attribute; it merely allows one to control the first few words and the length of the text. We show how to mitigate this undesirable effect by using simpler encoders and decoders, as well as pretraining schemes. These simpler variants learn to encode global features, such as topic and sentiment, in the latent representation more clearly. Moreover, these global aspects are more often reflected in the reconstructions that they produce.

In Chapter 9, we use autoencoders for studying language emergence. When artificial neural networks make mistakes, it is often said that they do not generalise *compositionally* (for example, in Lake and Baroni [2018]'s work). This term comes from formal semantics, where the meaning of a constituent is propagated upwards, following the syntactic tree [Montague, 1970, Szabó, 2020]. Formal semantics can correctly tell that the expression "blue car" denotes something that is both blue and a car. However, it cannot explain *why* we say "blue car" instead of alternatives such as "that", "a four-wheeled vehicle", or "a Honda Civic", alternatives that could also fit in a particular context. Language emergence simulations can help us answer such questions, questions involving language and psychology.

In language emergence simulations, language is the result of an optimisation process. In our case, the autoencoder represents two interlocutors. The encoder plays the role of a speaker. It observes some input and produces a message – a sequence of symbols. The decoder plays the role of a listener. It tries to reconstruct the input that the speaker has observed and talked about in its message. In our task, agents are asked to communicate about one or several entities. We analyze how they refer to several entities in a single sentence. Firstly, do they cleanly separate the information using one phrase per entity, as natural languages do? Secondly, when they talk about Mary and John, how do they differentiate between the meanings of "John loves Mary" and "Mary loves John"? We propose new metrics to answer these two questions in a fine-grained way. Moreover, we hypothesize that humans can separately refer to entities via different phrases thanks to their ability to conceive of these different entities as separate wholes. We implement this ability as an inductive bias. We find that indeed, agents that attend to objects wholistically talk more naturally, compared to agents which only observe a set of unrelated features belonging to different objects.

Before the articles, we present a brief primer on ML and deep learning in Chapter 2, followed by a review of modern NLP techniques in Chapter 3. After the articles, in Chapter 10, we summarize our results and discuss them in light of bigger questions and trends such as large language models and grounding.

Chapter 2

Deep learning

Machine learning (ML) algorithms adapt to the data that they are exposed to, in a process called *learning*. Current research in ML focuses on a growing number of tasks: autonomous driving, speech recognition and synthesis, medical diagnosis from images, sounds or blood samples, music recommendation, translating or summarizing documents, etc. These tasks are very diverse and involve different modalities such as texts, images, sounds, etc. Yet they can all be framed as learning tasks and solved using relatively similar ML techniques.

ML is based on statistics, with a heavy focus on predictive models. Moreover, ML is about learning *algorithms*. Algorithmic complexity is generally an important concern, since both the size of the dataset and the dimension of each example in a dataset can be very large.

Deep learning methods are particularly useful with such large datasets of high-dimensional data. Deep learning consists in training artificial neural networks with gradient-based methods. Given a dataset, deep learning practitioners proceed as follows: They specify a parametric model, a loss function, and then find the parameters of the model that minimize the loss function. In this chapter, we will describe in broad strokes how this is usually done.

2.1. A first introductive example

Let X be a random vector drawn from a fixed but unknown probability distribution p, defined over some space $\mathcal{X}$. Let $\mathcal{Y}$ be another space and f a function from $\mathcal{X}$ to $\mathcal{Y}$. Let $(x^{(1)}, \dots, x^{(n)})$ be a sample drawn i.i.d. from p and denote the *dataset* by $D = \{(x^{(1)}, f(x^{(1)})), \dots, (x^{(n)}, f(x^{(n)}))\}$.

In a *supervised learning* task, our goal is to predict the *output* or *target* $Y = f(X)$ given the *input* X. $\mathcal{X}$ and $\mathcal{Y}$ can be very different, so many real-world tasks can be cast in this framework. For example, we may want to determine if a fruit is healthy or unhealthy (Y) given a picture of the fruit (X); we may want to translate a text in English (X) to Chinese (Y); we may want to synthesize the waveform of a voice (Y) given information about a person's gender and age and a text to pronounce (X); etc.

A *parametric supervised learning model* (or *model* for short), denoted by f_θ, is a function from $\mathcal{X} \times \Theta$ to $\mathcal{Y}$. $\theta \in \Theta$ is a *parameter* vector. To start with, let us consider the *regression* case, where $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}^m$.

The *loss function* or *objective* quantifies the prediction errors made by a given model. Given a datapoint $(x,y) \in D$, it compares the prediction $\hat{y} = f_\theta(x)$ to the ground-truth label y. It is typically a non-negative function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$. When it reaches its minimum 0, the model is perfectly accurate. A popular loss function for regression is the *mean squared error* (MSE) loss:

$$l_{\text{MSE}}(\hat{y}, y) = ||\hat{y} - y||^2$$

The *classification* setting corresponds to $\mathcal{Y} = \{y_1, \ldots, y_k\}$. The 0-1 loss is often used in this setting:

$$l_{0-1}(\hat{y}, y) = \mathbb{1}(\hat{y} \neq y),$$

where $\mathbb{1}(c)$ is 1 if the condition c is true, else it is 0.

When we talk about a *task* to solve, it can be defined as the combination of a function f to approximate, some distribution over $\mathcal{X}$ and a loss function.

The simplest kind of model is *linear*, i.e. fully determined by a matrix W of size $m \times d$ and a *bias* vector $b \in \mathbb{R}^m$, defined by $\hat{y} = f_\theta(x) = Wx + b$.

We want to find the parameter $\theta^* \in \Theta$ that minimizes the *true error* or *true risk*

$$\mathcal{R}_{\text{true}}(f_\theta) = \mathbb{E}_{X \sim p}\, l(f_\theta(X), f(X)).$$

Indeed, if we find the parameter θ^* such that the true error is 0, we are able to predict Y given X perfectly for any X drawn from P. Models that achieve a low true risk are said to *generalise*, since they work well in the general case, on expectation.

However, the generalisation error is not known because the true distribution P is not known. Instead, we typically minimize the *empirical error* or *empirical risk*

$$\mathcal{R}_{\text{emp}}(f_\theta) = \sum_{(x,y) \in D} l(f_\theta(x), y).$$

This approach is known as *empirical risk minimisation* (ERM). When we use parametric models, we call *training*, *learning* or *optimisation* the process of iteratively modifying the parameter vector to find the one that minimizes some empirical risk.

If our model space $\{f_\theta | \theta \in \Theta\}$ contains f, statistical learning theory tells us that given a large enough dataset D, we can bound the true risk by an arbitrary scalar ϵ with some probability [Shalev-Shwartz and Ben-David, 2014, §2.3.1]. The number of datapoints necessary to achieve this risk ϵ at this level of certainty is called the *sample complexity*. While this theory does not seem applicable to the large model spaces of deep neural networks, the term is often used informally when comparing models trained with different amounts of data.

For a given parameter θ, the empirical risk is an unbiased estimate of the true risk. However, when θ is a random vector that depends on the sample D through the training process, then the empirical risk is not an unbiased estimate of the true risk anymore. That is why it is common practice to partition the sample D into a *train set* D_{train} and a *test set* D_{test}. The train set is used only for training and the test set only to evaluate the performance of the model.

2.2. Probabilistic models

2.2.1. Dealing with uncertainty

In the previous section, we assumed that the labels Y were related to the inputs through a function f. However, in most tasks, there is uncertainty as to what the true label Y is. This uncertainty can have several causes.

Consider for example the case of machine translation. Usually, the labels in a dataset are collected by one or several human annotators. Sometimes, there are several possible labels for a given input. For example, a single annotator might consider that several possible translations y correspond to a given document x. Moreover, even if each annotator has a strong opinion about a target y, y might differ from one annotator to the other. There can also be annotation mistakes due to a lack of knowledge from the annotator. Thus, for each x, there are often several corresponding labels y that are more or less appropriate.

This uncertainty is modeled by using a *joint* distribution over $\mathcal{X} \times \mathcal{Y}$, defined by a probability density function (regression case) or probability mass function (classification case) p. The deterministic case is still handled correctly in this formalism. Moreover, the statistical dependency between X and Y can be arbitrarily weak, as long as they are not independent.

In the remainder, we now think of the real data-generating process as non-deterministic. To capture this increased complexity, we switch to non-deterministic or *probabilistic* models (for a general reference, see Murphy [2012]). Such models do not predict a single value $\hat{y} \in \mathcal{Y}$ but instead produce a conditional density or probability p_θ over $\mathcal{Y}$ given $x \in \mathcal{X}$. The learning problem is now split into two problems:

- given the true conditional $p_{Y|X}$, what is the optimal prediction to minimize the empirical risk?
- given a parametric probabilistic model p_θ, how do we find θ such that p_θ approximates the conditional $p_{Y|X}$ well?

We answer these two questions in turn.

2.2.2. Bayes decision rule

Suppose that we are in the classification setting and care about minimizing the 0-1 loss. Given an input x, what is the optimal decision $\delta(x)$ that minimizes the 0-1 loss, on expectation with respect to the true distribution? This is called the *Bayes decision rule*:

$$\begin{aligned}\delta(x) &= \arg\min_{y^* \in \mathcal{Y}} \mathbb{E}_{p_{Y|X}}[\mathbb{1}(y^* \neq Y)|X = x] \\ &= \arg\min_{y^* \in \mathcal{Y}} \sum_{y \in \mathcal{Y}} p(y|x)\mathbb{1}(y^* \neq y) \\ &= \arg\min_{y^* \in \mathcal{Y}} (1 - p(y^*|x)) \\ &= \arg\max_{y^* \in \mathcal{Y}} p(y^*|x).\end{aligned}$$

It is optimal to choose the class y that has maximum conditional probability. Similar calculations show that in the regression case, given the true conditional probability, the optimal decision according to the MSE loss is the posterior mean $\delta(x) = \mathbb{E}_{p_{Y|X}}[Y|X = x]$.

We can also answer questions which were impossible to answer without modelling the joint distribution. For example, in the regression setting, we can compute the probability that Y lies in an interval of $\mathcal{Y}$. In the discrete setting, we can find the top-N classes, or measure the uncertainty over our predictions using the conditional entropy $H[Y|X = x] = -\sum_{y \in \mathcal{Y}} p(y|x)$.

For NLP problems involving text generation, $\mathcal{Y}$ is very large and we often resort to approximate search algorithms such as beam search.

2.2.3. Maximum likelihood estimation

We now turn to the second question: how do we find a parameter vector θ such that p_θ approximates $p_{Y|X}$ well? *Maximum likelihood estimation* (MLE) is one of the main methods to train probabilistic models (see for example Shalev-Shwartz and Ben-David [2014, Chap. 24]). The *likelihood* is defined as $\mathcal{L}(\theta|x) = p_\theta(x)$. As the name indicates, MLE consists in choosing the parameters that maximize the likelihood of the model.

In general, we maximize the log of the likelihood instead. This has two main benefits. Firstly, the probability of the entire dataset is a product of probabilities, since we assume that each datapoint is sampled independently from the true distribution. The log turns this product into a sum. Secondly, the most common probability distributions are in the *exponential family* and have a log-linear pmf or pdf.

MLE is ERM with the *negative log likelihood* (NLL) used as a loss:

$$\begin{aligned}\theta^* &= \arg\max_\theta \log \prod_{(x,y)\in D} p_\theta(y|x) \\ &= \arg\max_\theta \sum_{(x,y)\in D} \log p_\theta(y|x) \\ &= \arg\min_\theta \sum_{(x,y)\in D} -\log p_\theta(y|x).\end{aligned}$$

This loss is no longer a function on $\mathcal{Y} \times \mathcal{Y}$, but rather a function defined on $\mathcal{Y} \times \Delta^{k-1}$, where $K = |\mathcal{Y}|$ is the number of classes in our problem, and $\Delta^{K-1} = \{x \in [0,1]^K | \sum x_i = 1\}$. For each x, our model defines a random variable distributed categorically by outputting a probability vector from Δ^{K-1}. The NLL $-\log p_\theta(y|x)$ is minimized (0) when the probability of the target y is maximal (1). This loss is often called the *cross-entropy* loss.

In the regression case, we can choose to model the conditional distribution as a Gaussian with mean parameter depending on x and constant variance σ^2 for all x, $p_\theta(y|x) = \mathcal{N}(y|\mu(x), \sigma^2)$. In this case, we can show that MLE is equivalent to ERM with the MSE loss.

2.3. Layers and activation functions

In deep learning jargon, a *layer* is a function of inputs and learnable parameters, while *activation functions* are functions of inputs without learnable parameters. Layers and activation functions are composed to create *artificial neural networks*.

2.3.1. Multi-layer perceptron

A *multi-layer perceptron* (MLP) layer is made of several layers: linear transformations interlaced with non-linear activation functions. The simplest example is the 1-hidden-layer MLP. The 1-hidden-layer MLP has a preeminent theoretical role: roughly speaking, any mathematical function can be represented with an MLP [Hornik et al., 1989].

For a continuous input $x \in \mathbb{R}^{n_1}$, it produces a hidden representation $h \in \mathbb{R}^{n_2}$ and finally outputs $y \in \mathbb{R}^{n_3}$. Typical activation functions include tanh, the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and rectified linear units (ReLUs) defined by $g(x) = \max(0, x)$. When applied to vectors, an activation function such as σ or g is applied component by component. We denote one of these activation functions by g.

Given W_1 a $n_2 \times n_1$ real matrix, W_2 a real $n_3 \times n_2$ matrix, $b_1 \in \mathbb{R}^{n_2}$ and $b_2 \in \mathbb{R}^{n_3}$:

$$\begin{aligned}h &= g(W_1 x + b_1) \\ y &= W_2 h + b_2\end{aligned}$$

A layer is said to be *fully-connected* when all its outputs depend on all its inputs. For example, here, in the first layer, we have that $\forall i$, $\forall j$, h_i depends on x_j.

2.3.2. Softmax activation

The softmax activation is typically used to normalize an input vector, so that the outputs sum to 1. This is very useful to produce distributions over finite sets of elements. Therefore, unlike the tanh or ReLU activation, it is not applied component-wise. It is defined as a function $\mathbb{R}^d \to \mathbb{R}^d$ where the i-th component is equal to

$$\text{softmax}_i(x) = \frac{\exp(x_i)}{\sum_{j=1}^{d} \exp(x_j)}.$$

2.3.3. Attention mechanism

Machine translation models usually align each word in the translation y_i with a subset of words in the source text x [Brown et al., 1993]. In the context of machine translation, Bahdanau et al. [2014] introduced the *attention* mechanism to learn to align in a differentiable manner. An *attention mechanism* can be understood as a differentiable key-value database. A memory is a set of key-value vectors $M = \{(k_1, v_1), \ldots, (k_n, v_n)\}$ where $k_i \in \mathbb{R}^{n_1}$ and $v_i \in \mathbb{R}^{n_2}$. In practice, it is frequent to use identical vectors for both keys and values. For example, in machine translation, this memory can be a set of vectors representing the words in the source sentence x.

Given a query vector $q \in \mathbb{R}^{n_1}$, the relevance of the query to each memory slot $\tilde{\alpha}_i$ is measured by comparing each key to the query. This comparison can be implemented using the inner product $\tilde{\alpha}_i = q^\top k_i$, or using an MLP f with learned parameters, $\tilde{\alpha}_i = f([q; k_i])$ where $[\cdot;\cdot]$ denotes vector concatenation. These weights are usually normalized as $\alpha = \text{softmax}(\tilde{\alpha})$. A different query is produced at different step of the production of the output vector y, so that the different parts of the source are translated.

Once these attention weights α_i are computed, the convex combination of values is output:

$$y = \sum_{i=1}^{n} \alpha_i v_i.$$

2.4. Architectures for sequence modelling

An *architecture* is made of layers and activation functions that are composed. In this section we present architectures that are used to handle sequential data such as text. These models have in common that for a given input $x = (x_1, \ldots, x_{n_x})$ an input sequence of continuous vectors of $\mathbb{R}^d$, they produce a sequence of *hidden states* $h = (h_1, \ldots, h_{n_x})$ where each $h_i \in \mathcal{H} = \mathbb{R}^{d'}$.

A *Recurrent Neural Network* (RNN) [Jordan, 1997, Elman, 1990] is a neural network $f : \mathcal{H} \times \mathcal{X} \times \Theta \to \mathcal{H}$ defined by

$$h_t = f_\theta(h_{t-1}, x_t).$$

$h_t \in \mathcal{H}$ is the t-th hidden state and θ is the set of parameters of the RNN. t is often called a *timestep*, a legacy of the use of such models for time series. The initial hidden state h_0 is included in θ.

At each timestep t, a new hidden state is computed based on the previous timestep hidden state h_{t-1} and the current input x_t. Thus, the RNN subsumes information from the inputs seen so far $(x_1, ..., x_t)$ into the vector h_t. The content of h_t is determined during training and depends on the task, the architectural details of f_θ, the optimisation procedure, etc. RNNs have many use cases which will be discussed in the next chapter in the context of NLP.

In practice, RNNs are hard to train because of *vanishing/exploding gradients*. The *Long Short-Term Memory* (LSTM) architecture alleviates this problem to some degree [Hochreiter and Schmidhuber, 1997, Gers et al., 2000]. It became an essential part of the NLP toolkit, before being displaced by more powerful models. Greff et al. [2016] provides a useful empirical investigation into the importance of the many design choices of the LSTM.

Sutskever et al. [2014] proposed the *sequence-to-sequence* (*seq2seq*) architecture, where two RNNs are chained. The goal is to map an input $x = (x_1, \ldots, x_{n_x})$ to an output $y = (y_1, \ldots, y_{n_y})$, say when we map one text to another as in machine translation or summarisation. The first LSTM (the encoder) plays the role of a feature extractor, which processes an input sequence x and from which we keep the last hidden state h_{n_x}. The second LSTM (the decoder) is fed the vector h_{n_x} to decode the first element y_1. The information in this vector can of course be carried forward by the decoder if it is relevant. This way, the decoder is conditioned on information from the encoder.

This architecture creates a strong bottleneck: the entire input x should be summarized in a single vector h_{n_x}. The attention mechanism [Bahdanau et al., 2014] presented above can be integrated in the decoder LSTM to avoid this potentially problematic constraint. To do this, let the decoder produce an attention query q_t as a function of its hidden states h'_t at each timestep. Then, the attention mechanism computes a convex combination of $(h_1, \ldots, h_{n_x})$ as a function of the query q_t (using h_i as both keys and values). This allows the decoder to access different parts of the input sequence x at different places in the output sequence.

Various models were proposed based on attention mechanisms, among which the Transformer is by far the most successful [Vaswani et al., 2017] (see also Rush [2018] for the implementation). Transformers have almost entirely replaced LSTMs in NLP.

While LSTMs and Transformers can look rather cryptic, Levy et al. [2018] provide a very insightful comparison of these models. They strip the LSTM to its bare minimum while trying to keep a similar performance. What they find is that this stripped-down LSTM performs a

weighted sum of its (transformed) input vectors. In particular, the weights of past inputs can only decrease while the sequence is being processed, biasing the LSTM towards attending to nearby inputs. There is a similar bias in Transformers (due to the use of *positional embeddings* which are added to the input vectors), but it seems that it can be compensated. This partly explains why Transformers can deal with dependencies between elements in the inputs that are further away, compared to LSTMs.

2.5. Generalisation and regularisation

Sometimes, machine learning algorithms do not generalise as much as we want, i.e. when their true risk is too high for application of the model in the real world. There are two possible cases.

In the first case, the empirical risk is high as well. This is commonly referred to as *underfitting*. We often say that the model is not *expressive* or *powerful* enough, or it does not have enough *capacity*. Formally, this means that the mathematical function that we want to approximate (the true conditional pmf or pdf $p_{Y|X}$) cannot be approximated to a sufficient degree with models in the family $f_\theta|\theta \in \Theta$. Another possible explanation is that there is a problem with the optimisation procedure. Perhaps the parameters are not initialized properly, or the optimisation algorithm is not up to the task, resulting in suboptimal parameters.

In the second case, the empirical risk is low while the true risk is high, and we say that the model *overfits*. The model is too specific to the training data and does not extrapolate to datapoints outside of the dataset. Various cures exist against overfitting. They are usually called *regularizers*.

Some regularizers are additional terms in the loss function that is minimized. L1-regularization [Tibshirani, 1996] and L2-regularization [Hoerl and Kennard, 1970] are well-studied regularizers in the context of linear models. They consist in adding the term $||\theta||_1$ to our loss function (L1) or $||\theta||_2^2$ (L2). This encourages most θ_i to be close to 0, ensuring that only a few inputs explain the output.

Besides adding terms to the loss, we can also add noise at various points of the computations. Dropout [Hinton et al., 2012] is an operation that zeros out the components of a vector randomly with some small probability.

Early stopping [Sjöberg and Ljung, 1995] is a frequently used regularizer, which consists in stopping the optimisation before the minimum of the empirical loss is reached. It is common to put aside a subset of the training set, the *validation set*, on which we can estimate the true risk periodically. We can stop when it starts to increase.

2.6. Optimisation

When we perform linear regression, we can find the optimum θ^* analytically [Murphy, 2012, §7.3]. Deep learning models are complex functions for which this is not possible. However, they are designed to be differentiable so that we can perform *gradient descent* to optimize them. Starting from a random parameter vector θ, gradient descent iteratively updates θ by moving slightly down the direction of the gradient. More precisely, if we denote by $\theta^{(t)}$ the estimate of the parameter vector at timestep t, gradient descent computes the update

$$\theta^{(t+1)} \leftarrow \theta^{(t)} + \alpha \nabla_\theta \mathcal{R}_{\text{emp}},$$

where α is the *learning rate*, a parameter that should be sufficiently small for theoretical guarantees to hold, and where $\nabla_\theta \mathcal{R}_{\text{emp}}$ is the *gradient* of R_{emp} with respect to θ evaluated in $\theta^{(t)}$. This guarantees convergence towards a local minimum.

In practice, we do not optimise R_{emp} directly because it is costly to evaluate the loss on all the datapoints. It is more common to use *Stochastic Gradient Descent* (SGD) and compute the loss on a randomly selected example of D_{train}, or to compute the loss on a *minibatch*, a small subset of D_{train}.

The gradients are computed efficiently using a dynamic programming algorithm called *backpropagation* or *reverse-mode automatic differentiation* [Rumelhart et al., 1986, Goodfellow et al., 2016]. The backpropagation algorithm is composed of a forward pass and a backward pass. During the forward pass, the input is *propagated*, that is, the loss is computed and all the intermediate results a_i saved. During the backward pass, the gradients of the loss with respect to each parameter are computed using the chain rule.

2.7. Unsupervised learning

At its core, supervised learning is about learning a mapping between some input space $\mathcal{X}$ and some output space $\mathcal{Y}$ given some example pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Learning is said to be supervised, in the sense that the labels $y \in \mathcal{Y}$ are usually produced by human annotators, who teach the machine a desired behavior.

In the *unsupervised learning* setting, there are no labels. $\mathcal{X}$ is usually a high-dimensional space of images, texts, videos, sounds, etc. Generally speaking, unsupervised learning models try to represent the correlations between the components of X. They are used for two purposes, very often both.

Firstly, they can be used for *representation learning*. In this case, such models transform elements $x \in \mathcal{X}$ into $x' \in \mathcal{X}'$, where x' can be used downstream as input to supervised learning models. For example, *dimensionality reduction* methods such as Principal Component Analysis (PCA) (e.g., Murphy [2012]) lowers the dimensionality of the data. Working with x' can be an advantage in terms of statistical efficiency as well as computational and memory efficiency.

Secondly, they can be used as *generative models*, to produce new datapoints that look like those that are in the sample. Unconditional generation has limited use, but many such models can then be used for conditional generation. In conditional generation, we want to control aspects of the generated datapoints.

Autoencoders and autoregressive models are popular classes of models. They are both used for representation learning and generative modelling. Goodfellow et al. [2016, Chapter 20] describe many more models, while more recent innovations in contrastive learning and flow-based models are surveyed by Jaiswal et al. [2020] and Weng [2018] respectively.

2.7.1. Autoregressive models

The goal of probabilistic generative modelling is to approximate the distribution p of a random vector or sequence X over a space $\mathcal{X}$. Probabilistic generative models are usually trained like supervised probabilistic models with MLE, except that the *unconditional* log likelihood $\log p_\theta(x)$ is maximized on datapoints of $D = (x_1, \ldots, x_n)$.

We first decide on an order on the components of X. Let us assume that there is some natural order, due to temporal or spatial proximity $1, 2, \ldots, d$. This is often the case in time-series, images, texts, sound waves, amino-acids and base pairs in biology, etc. An *autoregressive* model is defined as $p_\theta(x) = p_\theta(x_1)p_\theta(x_2|x_1)\ldots p_\theta(x_d|x_1, \ldots, x_{d-1})$.

This order is first and foremost guided by what we want to do with the model. If we are modelling time-series to make predictions about the future, it is natural to use time to index this autoregressive model. Then, given (discrete) time t, we can make a prediction by using the conditional $p(x_{t+1}|x_1, \ldots, x_{t-1})$. But the choice of spatial or temporal order is also a good choice (if possible), since it is very often the case that the correlations are stronger locally than between further away components. RNNs, LSTMs and Transformers are often used as auto-regressive models.

2.7.2. Autoencoders

Autoencoders (also called *auto associators* in the connectionist literature [Rumelhart et al., 1986]) are made of two neural networks, an *encoder*, and a *decoder*. The encoder f transforms some datapoint x into a representation $f(x) = h \in \mathcal{H}$. The decoder maps that representation back into the input space, producing a *reconstruction* $x' = g(h)$ of the input. All encoders are trained to minimize a term $d(x, x')$, where d quantifies the discrepancy between the original datapoint x and its reconstruction x'.

We say that h is a representation of x, in the sense that the random variable $f(X)$ is predictive of X. Formally, we mean that the *mutual information* between X and the representation $f(X)$ is positive. Mutual information can be written as

$$I[X; f(X)] = \mathbb{E}[-\log p(X) - (-\log p(X|f(X)))].$$

It is the expected difference between two log losses: the loss incurred by the best model that predicts X unconditionally, i.e. without any information about X; and the loss incurred by the best possible model that predicts $f(X)$ given X.[1] Mutual information is graded: it quantifies how much information about x can be recovered from h.

There is a variety of models and different objectives. Many autoencoders reduce dimensionality by using a representation space that is smaller than $\mathcal{X}$. This is usually done for computational and statistical reasons: it makes downstream models faster and less prone to overfitting. Moreover, one concern is that if $\mathcal{H}$ is larger than $\mathcal{X}$, the autoencoder can learn the identity mapping. Contractive autoencoders [Rifai et al., 2011] do not suffer from this problem. They encourage the partial derivatives of h_i with respect to x_j to be close to 0, for all i and j. When this is the case, h_i does not change when x_h changes. The idea is to balance a code h that allows for a good reconstruction with a code h that does not vary wildly as a function of x.

The denoising autoencoder (DAE) [Vincent et al., 2008] is an autoencoder that is fed inputs that are artificially corrupted by noise, but which targets are the non-noisy, uncorrupted inputs. This encourages the learned representations to be invariant with respect to small perturbations of the input.

The variational autoencoder (VAE) [Kingma and Welling, 2013] is a Bayesian model made from three components. A prior probability distribution $p(h)$ is fixed over $\mathcal{H}$, with the parametric distribution of our choice. The encoder produces the parameters of an approximate posterior distribution, $q_\phi(h|x)$, while the decoder models the probability $p_\theta(x|h)$. Once the model is trained, we can sample from $p(h)p_\theta(x|h)$ to generate datapoints. The VAE is trained to minimize the sum of the reconstruction loss and the *Kullback-Leibler* divergence

$$D_{\mathrm{KL}}(q_\phi||p) = \sum_{h\in\mathcal{H}} \log q_\phi(h|x) \frac{\log q_\phi(h|x)}{\log p(h)}.$$

This term encourages the approximate posterior distributions to be close to the prior. To minimize the KL term, *reparametrisations* are employed, see for example Kingma and Welling [2013], Rezende et al. [2014]'s works for the Gaussian case and Jang et al. [2017], Maddison et al. [2016]'s papers for the categorical case.

Various methods were proposed to train autoencoders, such as recirculation [Hinton et al., 2012] and wake-sleep [Hinton et al., 1995], but backpropagation is still the most commonly used.

[1] It also has information-theoretic interpretations [Cover, 1999].

Chapter 3

Deep learning for natural language processing

Natural language processing (NLP) programs are *learned* – they are the result of applying deep learning techniques to text datasets. In this chapter, we go through some of the specificities of working with language. We start with a brief example of how deep learning techniques can be applied directly in the supervised learning setting. After that, we review the most important unsupervised learning techniques in NLP: segmentation algorithms, word and sentence embedding methods, and language models for directly solving downstream tasks.

3.1. Supervised learning tasks

Suppose that we have a document, represented as a sequence of n words. Each word is drawn from a finite vocabulary $\mathcal{V} = (w_1, \ldots, w_{|\mathcal{V}|})$. To simplify notation, we represent the document as a sequence of indices in the vocabulary, i.e. $x = (x_1, \ldots, x_{n_x})$ where $\forall i \in \{1, \ldots, n\}, x_i \in \{1, \ldots, |\mathcal{V}|\}$.

Various NLP tasks can be roughly classified into the following categories:

- *Regression* or *classification* tasks: the output is a single scalar or class, for example, a rating (regression) or the topic of a document (classification).
- *Sequence labelling* tasks: for each word x_i in the sequence, there is a target y_i, for example, a part of speech tag (determiner, verb, noun, etc.).
- *Text generation* tasks: the output is another document $y = (y_1, \ldots, y_{n_y})$, as in summarization, machine translation, etc. This is different than sequence labelling, since in general, $n_x \neq n_y$, and each output word y_i is correlated with all the words in the input x.

How do we use the sequence of words as inputs to deep learning models? The *one-hot encoding* of word w_i, ooh(w_i), is the vector of dimension $|\mathcal{V}|$ containing only zeroes, except for a one at position i. An *embedding matrix* W is a $|\mathcal{V}| \times d$ continuous matrix, where each row W_i is a dense vector representing the word w_i. It can be randomly initialized, or initialized via representations learned using unsupervised learning, as we will see below. By doing the

matrix-vector product of the embedding matrix by a one-hot encoding vector, we obtain the corresponding dense vector $W\text{ooh}(w_i) = W_i$. The word embedding matrix W is a learnable set of parameters that can be jointly optimized with the rest of the model.

Thus, once inputs are embedded, we can directly feed the sequence of vectors $(W_{x_1}, \ldots, W_{x_{n_x}})$ to sequence models such as RNNs, LSTMs or nowadays, preferably Transformer encoders. These models produce representations $H = (h_1, \ldots, h_{n_x})$.

These representations are used differently, depending on the type of task. For regression or classification tasks, this sequence of vectors can be aggregated by averaging or max-pooling. Then a MLP can predict a probability distribution from this vector. For sequence labelling tasks, we can directly use h_i as input to a predictor which will learn to map to y_i. Finally, for text generation, it is popular to use an attention-based autoregressive model like an LSTM or a Transformer decoder. Such a model predicts attention weights after the generation of every word, thus selecting variable information from the input x.

3.2. Segmentation

NLP deals with written languages. There are different types of writing systems across languages, and sometimes within a language (Japanese, for example). Alphabets, such as the Latin alphabet and Korean Hangul, denote consonants and vowels separately. Syllabaries and abugidas denote entire syllables, for example, the Ethiopic script and Devanagari. Abjads only mark the consonants, for example, the Arabic script. All these systems operate at some level of sound (phonemic). By contrast, in logographic systems such as Chinese Hanzi and the Japanese kanjis, a logogram usually denotes a morpheme rather than a sound.

In all these cases, written language can be represented as a sequence of discrete elements $s = (s_1, \ldots, s_m)$, $s_i \in \mathcal{S}$ that we can call *letters* for simplicity.

In the previous section, we assumed that we had access to words and that ML models used words as inputs, not letters. While this is a reasonable first choice when it is possible, it is not always optimal. In general, a *segmentation* algorithm is used to transform the sequence of letters $s = (s_1, \ldots, s_m)$ to a shorter sequence of *elements* $x = (x_1, \ldots, x_n)$ where x_i are taken from a vocabulary $\mathcal{V}$. There is a sweet spot between fine-grained (large n) and coarse-grained (small n) segmentations.

Let us first look at it from the point of view of statistical efficiency. *Bag-of-word* models are simple models which are invariant to the order of the elements x in the input sequence. Such simple models only work when the segmentation is at an appropriate granularity level. Consider a Naive Bayes classifier for topic classification. If the elements are too fine-grained, say, each letter is an element, then the classifier will have poor performance. Indeed, the distribution over letters is not very different from one topic to another topic. If we choose elements to be words, the classifier could work, since we expect the distribution over words

to be significantly different from one topic to another. If the elements are too coarse-grained, for instance, spanning entire sentences, we run into a problem of statistical estimation. In reasonably-sized datasets, most sentences are never sampled. Thus, if the counts are all zero for new examples (and the classifier smoothes counts, by adding ϵ to every count), the classifier will also do very poorly. Such elements that appear in the test set but not in the train set are called *out-of-vocabulary*. Equally problematic, some elements are observed only a handful of times, and this can yield spurious correlations.

A very fine-grained segmentation can be turned into an advantage given enough computational resources and enough data. Naive Bayes classifiers assume that each element is generated i.i.d. conditionnally on the topic (bag-of-word assumption). By contrast, more complex models like LSTMs and Transformers can take into account interactions between elements, and thus can operate at a finer-grained segmentation level. There are two potential downsides. Firstly, these models have more parameters, and thus may require more data to work as well as a simpler model operating at the word level. Secondly, using such models is more computationally intensive. Indeed, many models have linear or worse time and/or memory complexity in the number of elements in the sequence. For example, Transformers have quadratic memory complexity, and RNNs have linear time complexity. In summary, depending on our dataset size and our computational requirements, we can empirically experiment with different segmentations and find the best one.

There are several ways to perform segmentation. In many languages such as English, word and sentence markers are indicated by spaces and punctuation signs, so we can simply split the sequence into words. In some other languages like Chinese or Thai, however, word boundaries are not indicated so this solution requires more work – sometimes the use of preprocessing via another machine learning algorithm. Moreover, synthetic languages have a very large number of different words due to a very productive morphology, so a word-based segmentation would lead to many out-of-vocabulary words and other sparsity problems.

There are many data-driven algorithms. Perhaps one of the simplest was hinted by Harris [1954]. One could formalize it as a greedy algorithm, where the boundaries of the segments should be placed after a letter s_t if the uncertainty about s_{t+1} given $s_1, \ldots, s_t$ is locally maximal. Creutz and Lagus [2002] proposed two influential methods for unsupervised morpheme discovery. Nowadays, *byte pair encoding* (BPE) [Gage, 1994, Sennrich et al., 2016] is the most popular method. BPE operates bottom-up, building a vocabulary by aggregating longer and longer words, while the recursive splitting method proposed by Creutz and Lagus [2002] is top-down.

GPT2 [Radford et al., 2019] uses BPE in interesting ways. First, segments can cross word boundaries. Linguistically, such long segments can be justified. A *collocation* or *multi-word expression* is a sequence of words which meaning is not transparent, but somewhat arbitrary and non-compositional, perhaps due to metaphorical use ("grey matter", "kick the bucket"). It

could be beneficial to treat these as single elements. Furthermore, GPT2 operates directly at the byte-level, on UTF-8-encoded unicode characters. This greatly reduces their vocabulary size. Indeed, if one applies BPE naively, every element in $\mathcal{S}$ will be in the final vocabulary, resulting in very large vocabularies when there are hanzis or kanjis in the training data.

3.3. Unsupervised word representations

After segmentation, each element x in the vocabulary $\mathcal{V}$ is seen as an atomic symbol: an element in a set that has no obvious structure. Data-driven segmentation algorithms like Morfessor and BPE usually encourage such elements to be non-compositional, hopefully recovering linguistic morphemes which grammarians use as well as multi-word expressions. Each element is represented by a vector $W_x \in \mathbb{R}^d$, which can be learned along with the other model parameters via backpropagation. This vector will be tailored to solve a particular task.

There is, however, another way to learn vector representations without supervision. A representation of an element x_i is said to be *distributional* when it encodes information about the words that occur in its neighborhood or context [Harris, 1954]. For instance, since "eye-doctor" and "oculist" occur in almost similar contexts, they will have similar representations. Thus, these representations do reflect meaning to some extent.[1]

The first distributional word vectors were computed using document retrieval algorithms such as *latent semantic analysis* (LSA) in an unintended way. LSA starts by creating a count matrix of tokens (rows) appearing in a set of documents (columns) [Deerwester et al., 1990]. This matrix is then factorized using singular vector decomposition (SVD), yielding small vectors representing the documents in terms of the words that they contain. However, it was soon discovered [Schütze, 1992, Lund and Burgess, 1996] that these models are also useful to represent words as well. Instead of counting term-document, they counted term-term co-occurrences within a small moving window. The representations of interest became the word vector representations instead of the context representations, thus creating distributional word vectors. There is a rich literature about these methods, called *count-based* methods, based on counting co-occurrences in a matrix and then factorizing this matrix.

With the increasing popularity of neural models, *prediction-based* methods became more popular.[2] Notably, Mikolov et al. [2013a] introduced two extremely influential algorithms: the *skip-gram* algorithm and the *Continuous Bag-Of-Words* (*CBOW*) algorithm. For simplicity, we only present the more popular skip-gram algorithm. The skip-gram model predicts all the elements within a window of size $c \in \mathbb{N}$ around a target element x_t. The corresponding loss

[1]This type of representation was first proposed by Harris [1954]. The goal was to provide a method of description of the structure of natural languages, *distributional analysis*, that would be as rigorous and objective as possible. In particular, it did not require an understanding of the language by the analyst, or at least strived to evacuate questions of meaning. As a result, the method described by Harris [1954] is procedural in nature and resembles some modern NLP algorithms.

[2]Apparently, skip-gram became more popular due to a bug in a popular implementation [İrsoy et al., 2021].

is derived by approximating the loss

$$l_{\approx \text{sg}} = \frac{1}{n} \sum_{t=1}^{n} \sum_{-c \leq j \leq c, j \neq 0} \log p(x_{t+j}|x_t).$$

Here, the probability of the element x_j being in a window around x_i, $p(x_j|x_i)$, is proportional to $\exp({v_{x_i}}^\top v'_{x_j})$, where a word x is represented by two vectors: v_x, its *input* vector, and v'_x, its *output* vector.[3] We can then use v_x as the main representation of the word.

Due to the high computational cost of the computation of the gradients of the log probability (linear in the number of words in the vocabulary k), the preferred loss uses a technique called *negative sampling* [Mikolov et al., 2013b].

The methods which came to dominate are based on *language modelling*. Language modelling is the task of probabilistic modelling of text, or some approximation thereof. For example, auto-regressive language models factorize the distribution as $p(x) = p(x_1) \prod_{i=2}^{n} p(x_i|x_1, \ldots, x_{i-1})$. Bengio et al. [2003] introduced the first *neural* language model and proposed to learn word vectors via backpropagation. They could be used successfully as inputs to other neural networks, as shown by Collobert and Weston [2008] and Turian et al. [2010].

Word2vec and related methods such as GloVe [Pennington et al., 2014] were much more popular than language modelling approaches as they were much faster – they can be trained on CPU, using order of magnitudes more data. They were entirely displaced by language modelling approaches when such methods became relatively affordable, as we will discuss.

3.3.1. Intrinsic evaluation

Let us pause and discuss methods to analyze these vectors. It is well known that downstream tasks benefit from pretraining [Collobert and Weston, 2008, Turian et al., 2010]. But it is not clear what kind of information is necessary to solve a given downstream task. Performance differences also do not tell us about qualitative differences between two different embedding methods. Moreover, the classical lexical relations such as synonymy and antonymy, hypernymy and hyponymy (the *is a* relation), etc. were used by non-neural NLP algorithms. Do word embeddings somehow encode these relations as well? *Intrinsic* evaluation methods are designed to probe these embeddings in a task-agnostic fashion, in order to uncover their content.

Word2vec was initially evaluated on a set of analogy tasks [Mikolov et al., 2013c,a]. In a nutshell, simple arithmetic operations can be used to retrieve powerful syntactic and semantic relations. The authors famously found that the vector $v_{\text{king}} - v_{\text{man}} + v_{\text{woman}}$ is close to v_{queen}.

[3] With log-bilinear models like word2vec algorithms, we cannot use a single set of vectors to compute this probability, say with $p(x_j|x_i) = \exp({v_{x_i}}^\top v_{x_j})$, since the probability of a word w appearing in its own context should be low. I cannot find where I have read that.

However, Linzen [2016], Rogers et al. [2017] recommend not using analogy tasks for comparing different embeddings methods, as they suffer from many problems.

Another popular form of benchmark consists of a pair of words scored according to how strongly they are related in a particular way. Hill et al. [2014b] distinguish between *association* or *relatedness*, and *similarity*. The words in the pairs "coffee, cup" and "coffee, tea" are both highly associated or related. However, only "coffee" and "tea" are highly similar, in virtue of being hot beverages.

As noted by McRae et al. [2012], both association and similarity scores are composite metrics. That is, two pairs of words can be similar, but for different reasons. For instance, "coffee" and "beverage" are similar because coffee is kind of a beverage (hypernymy-hyponymy); "coffee" and "tea" are similar in that they share a hypernym, "beverage"; and "coffee" and "hot water" are similar because hot water is a necessary part of a coffee (meronymy-holonymy). But not all semantic relations increase similarity. While synonyms are very similar, antonyms are very dissimilar. SimLex-999 [Hill et al., 2014b] and SimVerb-3500 [Gerz et al., 2016] are designed to measure similarity. Benchmarks that measure association strength such as MEN [Bruni et al., 2012] and WS353 [Finkelstein et al., 2001b] were gathered using minimal and/or ambiguous annotation guidelines, so it is hard to define precisely what they measure and what is association.

The performance of various embeddings on association and similarity benchmarks relies on *cosine similarity*. It measures the similarity between two non-0 vectors v and w as

$$\text{cossim}(v, w) = \frac{< v, w >}{||v||||w||}.$$

It is 1 if and only if v is a linear combination of w.

To evaluate the embeddings on these benchmarks, the cosine similarity between all pairs is computed. Then, the Spearman correlation between this score and the ground-truth ratings is computed. A high correlation means that the notion (association or similarity) is well captured by the word vectors.

Yaghoobzadeh and Schütze [2016] argue convincingly that these benchmarks are limited. To paraphrase (or give a similar argument), it is mathematically impossible for cosine similarity to correlate with different lexical relations strengths. Yet it is possible to have different projections into smaller subspaces where cosine similarity correctly models each different lexical relation. This is also coherent with the view that similarity and relatedness are composite relations. They advocate for the use of classifiers to extract information from these vectors.

3.3.2. Contextualized word embeddings

Levy et al. [2015] argued that count-based methods do not perform so differently from prediction-based methods such as word2vec when their hyperparameters are tuned similarly (for example, by choosing similar context sizes, ignoring or downsampling certain words, etc.). It might not be entirely surprising since the algorithms are somewhat similar [Levy and Goldberg, 2014b]. In particular, these methods do not prioritize or differentiate between words in the context[4], nor do they model quantities involving the probabilities of three words (or more) co-occurring.

However, we can obtain significantly different vectors by using the skip-gram algorithm, but using a different definition of the context and the quantities to predict. Instead of considering the context to be the surrounding words, Levy and Goldberg [2014a] use words that are within a certain distance in the dependency tree of the sentence. Moreover, they also predict not only each word, but each dependency relation between the two words, which are given by the parser. The vectors thus produced better capture similarity [Hill et al., 2014b].

Neural language models [Bengio et al., 2003, Collobert and Weston, 2008] go beyond the simple objective of capturing pairwise statistical dependencies between nearby words. Thus, it was likely that they would encode finer distinctions in word vectors. However, the major realization was that the hidden representations of sequential models could be used directly as *contextualized* or *contextual* word embeddings. The first contextual embeddings employed the source sentence encoder of a neural machine translation model as a task-agnostic feature extractor [McCann et al., 2017]. Peters et al. [2018]'s ELMo made several crucial steps forward. First, they used a language model objective, allowing them to leverage large unlabelled corpora. Second, they used deep models (stacked BiLSTM) and found that the representations at different depths were useful for different tasks. For instance, part of speech tagging performance is better using the outputs of the first BiLSTM, while word sense disambiguation is more efficient using the second BiLSTM outputs.

BERT [Devlin et al., 2018], unlike ELMo, is based on a powerful Transformer architecture instead of stacked BiLSTMs. It uses an objective called *masked language modelling*. The Transformer is given a sequence where a fraction of the elements is replaced by a special, *mask* token. The objective is to reconstruct these masked elements. Thus, BERT is a denoising autoencoder with a specific reconstruction objective that penalizes only the corrupted parts of the inputs (the masked tokens).

[4]Lund and Burgess [1996]'s method is an exception: it assigns more importance to words that are closed to the target.

3.4. Unsupervised sentence representations

Word2vec reinforced the idea that pretraining on large corpora was very helpful. This led the community to look for equivalent methods to represent sentences. Unlike elements of the vocabulary, sentences need to be represented *compositionally* [Szabó, 2020], that is, as a function of the elements that they contain.

We can divide these methods on two axis. Firstly, on the computational axis, there are some very cheap methods, inspired by word2vec, which are very efficient log-bilinear models: paragraph vectors [Le and Mikolov, 2014], sent2vec [Pagliardini et al., 2018], FastSent [Hill et al., 2016a]. By contrast, other methods employed computationally-intensive LSTM encoders and decoders, such as sequential autoencoders [Dai and Le, 2015, Hill et al., 2016a, Bowman et al., 2016] and skip-thought vectors [Kiros et al., 2015].

Secondly, as Hill et al. [2016a] notes, while all these models compute sentence representations compositionally, only some of these models rely on the distributional hypothesis at the sentence level: skip-thought and FastSent create sentence representations that are predictive of the the words in the nearby sentences. By contrast, other models only represent the internal content of the sentences (the words), but not their context (nearby sentences).

3.5. Large language models as multi-task learners and few-shot learners

The need to represent sentences as single vectors has since been reconsidered. We can train sequential models directly on contextualized embeddings to solve downstream tasks. The language models can be further trained with backpropagation (*fine-tuning*), or have constant parameters (*frozen*).

While previous works used large language models as feature extractors, Radford et al. [2019]'s groundbreaking work showed that the GPT2 model can be used directly, without additional training. In order to produce the summary of a text "X", they sample from the autoregressive language model conditioned on the string "X TL;DR ". "TL;DR" is popular internet slang for "Too Long; Didn't Read", and is typically used to announce that one is going to summarize (just like "In summary"). They show that this approach works surprisingly well for summarization. Other patterns can be used to perform machine translation, question answering, etc. Thus, Radford et al. [2019] demonstrated that long-range correlations can be captured by very large, powerful models; and that this offers a powerful way to solve many tasks without supervision.

GPT3 [Brown et al., 2020] is a larger model, trained on more data. The authors claim that it is able to generalize to new tasks via conditioning in the few-shot, one-shot and zero-shot setting. Thus, not only can we use frequent patterns such as "TL;DR" to denote a particular

task, but we can also induce the network to learn new tasks. For example, suppose that we want to solve a supervised learning text generation task by training on a single pair (x, y) and we want to obtain a prediction for x'. We can condition on "x $\rightarrow$ y; x' $\rightarrow$ " and let the model generate the rest.

These very powerful abilities seem to be possible because natural language is reflexive enough to describe NLP tasks. On the one hand, the expression "TL;DR " can be seen as a simple, idiomatic discourse connective. But on the other hand, it denotes a very strong and particular type of correlation between the text that precedes and the text that follows it. We can complexify this expression and turn it into a whole sentence (such as "Let me try to rephrase this without the jargon, sticking to the most elementary constructions, in two short sentences: "). Such expressions do occur in corpora, and, as a result, language models which are powerful enough to model long-range dependencies such as Transformers, can able to perform such multi-task learning, meta-learning and probably continual learning, via conditioning.

Chapter 4

Prologue to first article: Auto-Encoding Dictionary Definitions into Consistent Word Embeddings

Abstract: Monolingual dictionaries are widespread and semantically rich resources. This paper presents a simple model that learns to compute word embeddings by processing dictionary definitions and trying to reconstruct them. It exploits the inherent recursivity of dictionaries by encouraging consistency between the representations it uses as inputs and the representations it produces as outputs. The resulting embeddings are shown to capture semantic similarity better than regular distributional methods and other dictionary-based methods. In addition, the method shows strong performance when trained exclusively on dictionary data and generalizes in one shot.

Bibliographical entry: Tom Bosc and Pascal Vincent. Auto-Encoding Dictionary Definitions into Consistent Word Embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1522–1532, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1181

4.1. Context

This paper builds on our previous work [Bahdanau et al., 2017]. We had shown that we can produce word embeddings tailored for a downstream task on the fly using external information (dictionary definitions and spelling). This was useful for rare words or even out-of-vocabulary words, for which the spelling information was an untapped resource, and for which a dictionary definition was sometimes also available.

As detailed in the paper, this algorithm has several interesting characteristics. It is a postprocessing method that can be run once and then be used by downstream models without additional computations. Compared to the previous paper, we focus on the different kinds of information encoded in distributional representations versus dictionary definitions.

I was particularly interested in the one-shot learning abilities of the model. Whereas distributional methods require seeing a few occurrences of each word in context to produce a representation, a single definition suffices here. This advantage should not be overstated, since few-shot learning is also possible to some extent with properly-tuned distributional algorithms [Herbelot and Baroni, 2017]. Moreover, this ability opened the door to continual learning. The representations that were produced could be integrated to the embedding matrix of the encoder. Indeed, the penalty encourages the representations produced by the encoder to be usable as inputs.

Finally, distributional representations are inherently circular: they encode information about a token that is predictive of the presence of other nearby tokens. Dictionary definitions also have this circular quality. Despite this, the definition embeddings are not distributional. Rather, the consistency penalty is based on the idea of substitutability in linguistics. It was satisfying to compare these methods which both learn from data (the dictionary autoencoder, however, on *slightly* structured data structured as pairs).

4.1.1. Modifications to the published article

Following Tim O'Donnell's comments, the article presented here slightly differs from the published version. Some unclear abbreviations are now defined, model selection and hyperparameter search is presented more clearly, and a note was added to discuss weight tying with output embeddings.

4.2. Personal contributions

I designed the model, ran the experiments, and did most of the writing. I only did a small amount of programming, since most of the code was reused from Bahdanau et al. [2017]'s work and written by Dima. Pascal helped writing the paper and the rebuttals, and he provided high-level guidance throughout the project.

4.3. Contributions

- Introduction of an autoencoder variant, the *consistency penalized autoencoder* (CPAE) and a corresponding auxiliary loss.
- Empirical validation of the model.

Chapter 5

Auto-Encoding Dictionary Definitions into Consistent Word Embeddings

5.1. Introduction

Dense, low-dimensional, real-valued vector representations of words known as *word embeddings* have proven very useful for NLP tasks [Turian et al., 2010]. They can be learned as a by-product of solving a particular task [Collobert et al., 2011]. Alternatively, one can pretrain generic embeddings based on co-occurrence counts or using an unsupervised criterion such as predicting nearby words [Bengio et al., 2003, Mikolov et al., 2013b]. These methods implicitly rely on the distributional hypothesis [Harris, 1954, Sahlgren, 2008], which states that words that occur in similar contexts tend to have similar meanings.

It is common to study the relationships captured by word representations in terms of either *similarity* or *relatedness* [Hill et al., 2016c]. "Coffee" is related to "cup" as coffee is a beverage often drunk in a cup, but "coffee" is not similar to "cup" in that coffee is a beverage and cup is a container. Methods relying on the distributional hypothesis often capture relatedness very well, reaching human performance, but fare worse in capturing similarity and especially in distinguishing it from relatedness [Hill et al., 2016c].

It is useful to specialize word embeddings to focus on either relation in order to improve performance on specific downstream tasks. For instance, Kiela et al. [2015] report that improvements on relatedness benchmarks also yield improvements on document classification. In the other direction, embeddings learned by neural machine translation models capture similarity better than distributional unsupervised objectives [Hill et al., 2014a].

There is a wealth of methods that postprocess embeddings to improve or specialize them, such as retrofitting [Faruqui et al., 2014]. On similarity benchmarks, they are able to reach correlation coefficients close to inter-annotator agreements. But these methods rely on additional resources such as paraphrase databases [Wieting et al., 2016] or graphs of lexical relations such as synonymy, hypernymy, and their converse [Mrkšić et al., 2017].

Rather than relying on such curated lexical resources that are not readily available for the majority of languages, we propose a method capable of improving embeddings by leveraging the more common resource of monolingual dictionaries.[1] Lexical databases such as WordNet [Fellbaum, 1998] are often built from dictionary definitions, as was proposed earlier by Amsler [1980]. We propose to bypass the process of explicitly building a lexical database – during which information is structured but information is also lost – and instead directly use its detailed source: dictionary definitions. The goal is to obtain better representations for more languages with less effort.

The ability to process new definitions is also desirable for future natural language understanding systems. In a dialogue, a human might want to explain a new term by explaining it in his own words, and the chatbot should understand it. Similarly, question-answering systems should also be able to grasp definitions of technical terms that often occur in scientific writing.

We expect the embedding of a word to represent its meaning compactly. For interpretability purposes, it would be desirable to be able to generate a definition from that embedding, as a way to verify what information it captured. Case in point: to analyse word embeddings, Noraset et al. [2017] used RNNs to produce definitions from pretrained embeddings, manually annotated the errors in the generated definitions, and found out that more than half of the wrong definitions fit either the antonyms of the defined words, their hypernyms, or related but different words. This points in the same direction as the results of intrinsic evaluations of word embeddings: lexical relationships such as lexical entailment, similarity and relatedness are conflated in these embeddings. It also suggests a new criterion for evaluating word representations, or even learning them: they should contain the necessary information to reproduce their definition (to some degree). In this work, we propose a simple model that exploits this criterion. The model consists of a definition autoencoder: an LSTM processes the definition of a word to yield its corresponding word embedding. Given this embedding, the decoder attempts to reconstruct the bag-of-words representation of the definition. Importantly, to address and leverage the recursive nature of dictionaries – the fact that words that are used inside a definition have their own associated definition – we train this model with a consistency penalty that ensures proximity of the embeddings produced by the LSTM and those that are used by the LSTM.

Our approach is self-contained: it yields good representations when trained on nothing but dictionary data. Alternatively, it can also leverage existing word embeddings and is then especially apt at specializing them for the similarity relation. Finally, it is also extremely data-efficient, as it permits to create representations of new words in one shot from a short definition.

[1]See Appendix A.1 for a list of online monolingual dictionaries.

5.2. Model

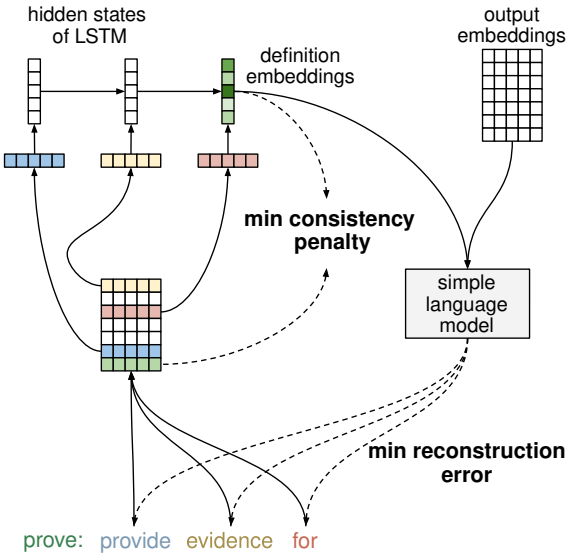

Figure 1. Overview of the CPAE model.

5.2.1. Setting and motivation

We suppose that we have access to a dictionary that maps words to one or several definitions. Definitions themselves are sequences of words. Our training criterion is built on the following principle: we want the model to be able to recover the definition from which the representation was built. This objective should produce similar embeddings for words which have similar definitions. Our hypothesis is that this will help capture semantic similarity, as opposed to relatedness. Reusing the previous example, "coffee" and "cup" should get different representations in virtue of having very different definitions, while "coffee" and "tea" should get similar representations as they are both defined as beverages and plants.

We chose to compute a single embedding per word in order to avoid having to disambiguate word senses. Indeed, word sense disambiguation remains a challenging open problem with mixed success on downstream task applications [Navigli, 2009]. Also, recent papers have shown that a single word vector can capture polysemy and that having several vectors per word is not strictly necessary [Li and Jurafsky, 2015, Yaghoobzadeh and Schütze, 2016]. Thus, when a word has several definitions, we concatenate them to produce a single embedding.

5.2.2. Autoencoder model

Let $\mathcal{V}^{\mathcal{D}}$ be the set of all words that are *used* in definitions and $\mathcal{V}^{\mathcal{K}}$ the set of all words that are *defined*. We let $w \in \mathcal{V}^{\mathcal{K}}$ be a word and $D_w = (D_{w,1}, \ldots, D_{w,T})$ be its definition, where

$D_{w,t}$ is the index of a word in vocabulary $\mathcal{V}^{\mathcal{D}}$. We encode such a definition D_w by processing it with an LSTM [Hochreiter and Schmidhuber, 1997].

The LSTM is parameterized by Ω and a matrix E of size $|\mathcal{V}^{\mathcal{D}}| \times m$, whose i^{th} row E_i contains an m-dimensional *input embedding* for the i^{th} word of $\mathcal{V}^{\mathcal{D}}$. These input embeddings can either be learned by the model or be fixed to a pretrained embedding. The last hidden state computed by this LSTM is then transformed linearly to yield an m-dimensional *definition embedding* h. Thus the encoder whose parameters are $\theta = \{E,\Omega,W,b\}$ computes this embedding h as

$$h = f_\theta(D_w) = W\,\mathrm{LSTM}_{E,\Omega}(D_w) + b.$$

The subsequent decoder can be seen as a conditional language model trained by maximum likelihood to regenerate definition D_w given definition embedding $h = f_\theta(D_w)$. We use a simple conditional unigram model with a linear parametrization $\theta' = \{E', b'\}$ where E' is a $|\mathcal{V}^{\mathcal{D}}| \times m$ matrix and b' is a bias vector.[2]

We maximize the log-probability of definition D_w under that model:

$$\begin{aligned} \log p_{\theta'}(D_w|h) &= \sum_t \log p_{\theta'}(D_{w,t}|h) \\ &= \sum_t \log \frac{e^{\left\langle E'_{D_{w,t}}, h\right\rangle + b'_{D_{w,t}}}}{\sum_k e^{\left\langle E'_k, h\right\rangle + b'_k}} \end{aligned} \tag{5.2.1}$$

where $\langle,\rangle$ denotes an ordinary dot product. We call E' the *output embedding* matrix. The basic autoencoder training objective to minimize over the dictionary can then be formulated as

$$J_r(\theta', \theta) = -\sum_{w \in \mathcal{V}^{\mathcal{K}}} \log p_{\theta'}(D_w|f_\theta(D_w)).$$

5.2.3. Consistency penalty

We introduced 3 different embeddings: a) *definition embeddings* h, produced by the definition encoder, are the embeddings we are ultimately interested in computing; b) *input embeddings* E are used by the encoder as inputs; c) *output embeddings* E' are compared to definition embeddings to yield a probability distribution over the words in the definition. We propose a soft weight-tying scheme that brings the input embeddings closer to the definition embeddings. We call this term a *consistency* penalty because its goal is to to ensure that the embeddings used by the encoder (input embeddings) and the embeddings produced by the encoder (definition embeddings) are consistent with each other. It is implemented as

[2]We have tried using a LSTM decoder but it didn't yield good representations. It might overfit because the set of dictionary definitions is small. Also, using teacher forcing, we condition on ground-truth words, making it easier to predict the next words. More work is needed to address these issues.

$$J_p(\theta) = \sum_{w \in \mathcal{V}^{\mathcal{D}} \cap \mathcal{V}^{\mathcal{K}}} d(E_w, f_\theta(D_w))$$

where d is a distance. In our experiments, we choose d to be the Euclidian distance. The penalty is only applied to some words because $\mathcal{V}^{\mathcal{D}} \neq \mathcal{V}^{\mathcal{K}}$. Indeed, some words are defined but are not used in definitions and some words are used in definitions but not defined. In particular, inflected words are not defined. To balance the two terms, we introduce two hyperparameters $\lambda, \alpha \geq 0$ and the complete objective is

$$J(\theta', \theta) = \alpha J_r(\theta', \theta) + \lambda J_p(\theta).$$

We call the model *CPAE*, for *Consistency Penalized AutoEncoder* when $\alpha > 0$ and $\lambda > 0$ (see Figure 1).[3]

The consistency penalty is a cheap proxy for dealing with the circularity found in dictionary definitions. We want the embeddings of the words in definitions to be compositionally built from their definition as well. The recursive process of fetching definitions of words in definitions does not terminate, because all words are defined using other words. To counter that, our model uses input embeddings that are brought closer to definition embeddings and vice versa in an asynchronous manner.

Moreover, if λ is chosen large enough, then $E_w \approx f_\theta(D_w)$ after optimisation. This means that the definition embedding for w is close enough to the corresponding input embedding to be used by the encoder for producing other definition embeddings for other words. In that case, the model could enrich its vocabulary by computing embeddings for new words and consistently reusing them as inputs for defining other words.

Finally, the consistency penalty can be used to leverage pretrained embeddings and bootstrap the learning process. For that purpose, the encoder's input embeddings E can be fixed to pretrained embeddings. These provide targets to the encoder but also helps the encoder to produce better definition embeddings in virtue of using input embeddings that already contain meaningful information.

To summarize, the consistency penalty has several motivations. Firstly, it deals with the fact that the recursive process of building representation of words out of definitions does not terminate. Secondly, it is a way to enrich the vocabulary with new words dynamically. Finally, it is a way to integrate prior knowledge in the form of pretrained embeddings.

In general, we might also want to further tie output embeddings with input and definition embeddings. However, this is impractical due to the simple decoder used here, for a similar reason as given by Goldberg and Levy [2014, footnote 2]. The log probabilities of words in a definition are given by the dot product of the definition embedding and the output embeddings. If they were tied, the log probability that any defined word appears in its own

[3]Our implementation is available at `https://github.com/tombosc/cpae`

definition would be high. However, defined words are mostly or always absent from their own definitions.

In order to study the two terms of the objective in isolation, we introduce two special cases. When $\lambda = 0$ and $\alpha > 0$, the model reduces to AE for *Autoencoder*. When $\alpha = 0$ and $\lambda > 0$, we retrieve *Hill's model*, as presented by Hill et al. [2016b].[4] Hill's model is simply a recurrent encoder that uses pretrained embeddings as targets so it only makes sense in the case we use fixed pretrained embeddings.

5.3. Related work

5.3.1. Extracting lexical knowledge from dictionaries

There is a long history of attempts to extract and structure the knowledge contained in dictionaries. Amsler [1980] studies the possibility of automatically building taxonomies out of dictionaries, relying on the syntactic and lexical regularities that definitions display. One relation is particularly straightforward to identify: it is the *is a* relation that translates to hypernymy. Dictionary definitions often contain a *genus* which is the hypernym of the defined word, as well as a *differentia* which differentiates the hypernym from the defined word. For example, the word "hostage" is defined as "a prisoner who is held by one party to insure that another party will meet specified terms", where "prisoner" is the genus and the rest is the differentia.

To extract such relations, early works by Chodorow et al. [1985] and Calzolari [1984] use string matching heuristics. Binot and Jensen [1987] operate at the syntactic parse level to detect these relations. Whether based on the string representation or the parse tree of a definition, these rule-based systems have helped to create large lexical databases. We aim to reduce the manual labor involved in designing the rules and directly obtaining representations from raw definitions.

5.3.2. Improving word embeddings using lexical resources

Postprocessing methods for word embeddings use lexical resources to improve already trained word embeddings irrespective of how they were obtained. When it is used with fixed pretrained embeddings, our method can be seen as a postprocessing method.

Postprocessing methods typically have two terms for trading off conservation of distributional information that is brought by the original vectors with the new information from lexical

[4]It is not exactly their model as we use Euclidian distance instead of the cosine distance or the ranking loss. They also explore several variants where the input embeddings are learned, which we didn't find to produce any improvement. We haven't experimented with the ranking loss, but the cosine distance does not seem to improve over Euclidian. Finally, they also use a simple encoder that averages word vectors, which we found to be inferior.

resources. There are two main ways to preserve distributional information: Attract-Repel [Vulić and Mrkšić, 2017], retrofitting [Mrkšić et al., 2017] and our method control the distance between the original vector and the postprocessed vector so that the new vector does not drift too far away from the original vector. Counter-Fitting [Mrkšić et al., 2016] and dict2vec [Tissier et al., 2017] ensure that the neighbourhood of a vector in the original space is roughly the same as the neighbourhood in the new space.

Finally, methods differ by the nature of the lexical resources they use. To our knowledge, dict2vec is the only technique that uses dictionaries. Other postprocessing methods use various data from WordNet: sets of synonyms and sometimes antonyms, hypernyms, and hyponyms. For instance, Lexical Entailment Attract-Repel (LEAR) uses all of these [Vulić and Mrkšić, 2017]. Other methods rely on paraphrase databases [Wieting et al., 2016].

5.3.3. Dictionaries and word embeddings

We now turn to the most relevant works that involve dictionaries and word embeddings.

Dict2vec [Tissier et al., 2017] combines the word2vec skip-gram objective (predicting all the words that appear in the context of a target word) with a cost for predicting related words. These related words either form *strong pairs* or *weak pairs* with the target word. Strong pairs have a greater influence in the cost. They are pairs of words that are in the neighbourhood of the target word in the original embedding, as well as pairs of words for which the definitions make reference to each other. Weak pairs are pairs of words where only one word appears in the definition of the other. Unlike dict2vec, our method can be used as either a standalone or a postprocessing method (when used with pretrained embeddings). It also focuses on handling and leveraging the recursivity found in dictionary definitions with the consistency penalty whereas dict2vec ignores this aspect of the structure of dictionaries.

Besides dict2vec, Hill et al. [2016b] train neural language models to predict a pretrained word embedding given a definition. Their goal was to learn a general-purpose sentence encoder useful for downstream tasks. Noraset et al. [2017] propose the task of generating definitions based on word embeddings for interpretability purposes. Our model unifies these two approaches into an autoencoder. However, we have a different goal: that of creating or improving word representations. Their methods assume that pretrained embeddings are available to provide either targets or inputs, whereas our model is unsupervised, and the use of pretrained embeddings is optional.

Bahdanau et al. [2017] present a related model that produces embeddings from definitions such that it improves performance on a downstream task. By contrast our approach is used either stand-alone or as as a postprocessing step, to produce general-purpose embeddings at a lesser computational cost. The core novelty is the way we leverage the recursive structure of dictionaries.

Finally, Herbelot and Baroni [2017] also aim at learning representations for word embeddings in a few shots. The method consists of fine-tuning word2vec hyperparameters and can learn in one or several passes, but it is not specifically designed to handle dictionary definitions.

5.4. Experiments

5.4.1. Setup

We experiment on English to benefit from the many evaluation benchmarks available. The dictionary we use is that of WordNet Fellbaum [1998]. WordNet contains graphs of linguistic relations such as synonymy, antonymy, hyponymy, etc. but also definitions. We emphasize that our method trains exclusively on the definitions and is thus applicable to any electronic dictionary.

However, in order to *evaluate* the quality of embeddings on unseen definitions, WordNet relations comes in handy: we use the sets of synonyms to split the dictionary into a train set and a test set, as explained in Section 5.7. Moreover, WordNet has a wide coverage and high quality, so we do not need to aggregate several dictionaries as done by Tissier et al. [2017]. Finally, WordNet is explicitly made available for research purposes, therefore we avoid technical and legal difficulties associated with crawling proprietary online dictionaries.

We do not include part of speech tags that go with definitions. WordNet does not contain function words but contains homonyms of function words. We filter these out.

5.4.2. Similarity and relatedness benchmarks

Evaluating the learned representations is a complex issue [Faruqui et al., 2016]. Indeed, different evaluation methods yield different rankings of embeddings: there is no single embedding that outperforms others on all tasks [Schnabel et al., 2015] and thus no single best evaluation method.

We leave aside analogy prediction benchmarks as they suffer from many problems [Linzen, 2016, Rogers et al., 2017]. We focus on intrinsic evaluation methods. In particular, we study how different models trade off similarity and relatedness. The benchmarks consist of pairs of words scored according to some criteria by human annotators. We score each pair by computing the cosine similarity between the corresponding word vectors. Then the predicted scores and the ground truth are ranked and the correlation between the ranks is measured by Spearman's ρ. The benchmarks vary in terms of annotation guidelines, number of annotators, selection of the words, etc.

As proposed by Faruqui et al. [2016], we use separate datasets for model selection. All algorithms have hyperparameters. For each algorithm, we perform a grid search to choose

the hyperparameters that perform best on the development sets of SimVerb3500 [Gerz et al., 2016] and MEN [Bruni et al., 2014], which are the only benchmarks with a standard development/test split. The space of hyperparameters searched is described in Appendix A.3. We justified our emphasis on the similarity relation in Section 5.1: capturing this relation remains a challenge, and we hypothesize that dictionary data should improve representations in that respect. The model selection procedure reflects that we want embeddings specialized in similarity: the validation score is a weighted mean where SimVerb weighs twice as much as MEN.

5.4.3. Baselines

The objective function presented in section 5.2 gives us 3 different models: CPAE, AE, and Hill's model. The objective of CPAE comprises the sum of the objective of Hill's model and of AE. We compare the CPAE model to both of these to evaluate the individual contribution of the two terms to the performance. In addition, when we use external corpora to pretrain embeddings, we compare these models to dict2vec and retrofitting.

The test benchmarks for the similarity relation includes SimLex999 [Hill et al., 2016c] and more particularly SimLex333, a challenging subset of SimLex999 which contains only highly related pairs but in which similarity scores vary a lot. For relatedness, we use MEN [Bruni et al., 2014], RG [Rubenstein and Goodenough, 1965], WS353 [Finkelstein et al., 2001a], SCWS [Huang et al., 2012], and MTurk [Radinsky et al., 2011, Halawi et al., 2012]. The evaluation is carried out by a modified version of the Word Embeddings Benchmarks project.[5] Conveniently, all these benchmarks contain mostly lemmas, so we do not suffer too much from the problem of missing words.[6]

5.5. Results in the dictionary-only setting

In the first evaluation round, we train models only using a single monolingual dictionary. This allows us to check our hypothesis that dictionaries contain information for capturing the similarity relation between words.

Our baselines are regular distributional models: GloVe [Pennington et al., 2014] and word2vec [Mikolov et al., 2013b]. They are trained on the concatenation of defined words with their definitions.

Such a formatting introduces spurious co-occurrences that do not otherwise appear in free text. But these baselines are not designed for dictionaries and cannot deal with their particular structure.

[5]Original project available at `https://github.com/kudkudak/word-embeddings-benchmarks`, modified version distributed with our code.
[6]Missing words are not removed from the dataset, but they are assigned a null vector.

We compare these models to the autoencoder model without (AE) and with (CPAE) the consistency penalty. In this setting, we cannot use Hill's model as it requires pretrained embeddings as targets. We also trained an additional CPAE model with pretrained word2vec embeddings trained on the concatenated definitions. The results are presented in Table 1.

	Development		Similarity			Relatedness				
	SV-d	MEN-d	SL999	SL333	SV-t	RG	SCWS	MEN-t	MT	353
GloVe	12.0	54.8	19.8	-9.1	7.8	57.5	46.8	57.0	49.4	44.4
word2vec	35.2	62.3	34.5	16.0	36.4	65.7	54.5	59.9	56.1	**61.9**
AE	34.9	42.7	35.6	26.8	32.5	64.8	50.2	42.2	38.6	41.4
CPAE ($\lambda = 8$)	42.8	48.5	39.5	29.1	34.8	67.1	54.3	49.2	42.6	48.7
CPAE-P ($\lambda = 64$)	**44.1**	**65.1**	**45.8**	**30.9**	**42.3**	**72.0**	**60.4**	**63.8**	**61.5**	61.3

Table 1. Positive effect of the consistency penalty and word2vec pretraining. Spearman's correlation coefficient $\rho \times 100$ on benchmarks. Without pretraining, autoencoders (AE and CPAE) improve on similarity benchmarks while capturing less relatedness than distributional methods. The consistency penalty (CPAE) helps even without pretrained targets. Our method, combined with pretrained embeddings on the same dictionary data (CPAE-P), significantly improves on every benchmark. Abbreviations: SV: SimVerb, SL: SimLex, 353: WS353, -d: development set, -t: test set.

GloVe is outperformed by word2vec by a large margin so we ignore this model in later experiments. Word2vec captures more relatedness than CPAE (+10.7 on MEN-test, +13.5 on MTurk, +13.2 on WS353) but less similarity than CPAE. The difference in the nature of the relations captured is exemplified by the scores on SimLex333. This subset of SimLex999 focuses on pairs of words that are very related but that can be either similar or dissimilar. On this subset, CPAE fares better than word2vec (+13.1).

The consistency penalty improves performance on every dataset. This penalty provides targets to the encoder, but these targets are themselves learned and change during the learning process. The exact dynamics of the system are unknown. It can be seen as a regularizer because it puts strong weight-sharing constraints on both types of embeddings. It also resembles *bootstrapping* in reinforcement learning, which consists of building estimates of values functions on top of over estimates [Sutton and Barto, 1998].

The last model is the CPAE model that uses the word2vec embeddings pretrained on the dictionary data. This combination not only equals other models on some benchmarks but outperforms them, sometimes by a large margin (+6.3 on SimLex999 and +7.5 on SimVerb3500 compared to CPAE, +6.1 on SCWS, +5.4 on MTurk compared to word2vec). Thus, the two kinds of algorithms are complementary through the different relationships that they capture best. The pretraining helps in two different ways, by providing quality input embeddings and targets to the encoder. The pretrained word2vec targets are already remarkably good. That is why the chosen consistency penalty coefficient selected is very high

($\lambda = 64$). The model can pay a small cost and deviate from the targets in order to encode information about the definitions.

To sum up, dictionary data contains a lot of data relevant to modeling the similarity relationship. Autoencoder based models learn different relationships than regular distributional methods. The consistency penalty is a very helpful prior and regularizer for dictionary data, as it always helps, regardless of what relationship we focus on. Finally, our model can drastically improve embeddings that were trained on the same data but with a different algorithm.

5.6. Improving pretrained embeddings

We have seen that CPAE with pretraining is very efficient. But does this result generalizes to other kind of pretraining data? To answer this question, we experiment using embeddings pretrained on the first 50 million tokens of a Wikipedia dump, as well as the entire Wikipedia dump. We compare our method to existing postprocessing methods such as dict2vec and retrofitting, which also aims at improving embeddings with external lexical resources.

Retrofitting, which operates on graphs, is not tailored for dictionary data, which consists in pairs of words along with their definitions. We build a graph where nodes are words and edges between nodes correspond to the presence of one of the words into the definition of another. Obviously, we lose word order in the process.

	Development		Similarity			Relatedness				
	SV-d	MEN-d	SL999	SL333	SV-t	RG	SCWS	MEN-t	MT	353
word2vec	21.7	71.1	33.2	6.9	21.2	68.5	**65.8**	71.5	61.1	**65.3**
retrofitting	28.5	**71.6**	36.9	14.1	26.1	**78.9**	65.7	**74.4**	**62.8**	60.7
dict2vec	26.3	63.5	36.2	15.5	22.0	69.2	63.8	63.9	54.9	60.8
Hill	26.9	63.3	27.7	12.9	21.7	72.9	58.4	64.1	54.0	52.3
AE	33.5	47.0	33.1	20.4	32.5	66.0	52.0	46.4	40.2	43.7
CPAE ($\lambda = 4$)	**39.5**	60.8	**39.9**	**26.6**	**37.8**	69.7	59.2	60.8	48.4	55.6

Table 2. Improving pretrained embeddings computed on a small corpus. Spearman's correlation coefficient $\rho \times 100$ on benchmarks. All methods use pretrained embeddings. All methods (except maybe Hill) manage to improve the embeddings. Retrofitting outperforms dict2vec and efficiently specializes for relatedness. CPAE outperforms AE and allows to trade off relatedness for similarity. Abbreviations: SV: SimVerb, SL: SimLex, 353: WS353, -d: development set, -t: test set.

The results for the small corpus are presented in Table 2. By comparing Table 2 with Table 1, we see that word2vec does worse on similarity than when trained on dictionary data, but better on relatedness. Both dict2vec and retrofitting improve with regards to word2vec on similarity benchmarks and seem roughly on par. However, dict2vec fails to improve on

	Similarity			Relatedness			
	SimLex999	SimLex333	SimVerb-test	WS353	MEN-test	SCWS	MTurk
All	36.5	27.7	36.9	43.7	48.8	53.2	41.5
Train	40.0	28.5	36.7	46.9	53.4	57.1	42.9
Test	27.5	25.4	38.5	42.5	44.1	40.3	39.1

Table 3. One pass generalisation. Spearman's correlation coefficient $\rho \times 100$ on benchmarks. The model is CPAE (without pretrained embeddings). All: all pairs in the benchmarks. Train: pairs for which both words are in the training or validation set. Test: pairs which contain at least one word in the test set. Correlation is lower for test pairs but remains strong ($\rho > 0.3$): the model has good generalisation abilities.

relatedness benchmarks, whereas retrofitting sometimes improves (as in RG, MEN, and MTurk), sometimes equals (SCWS) and does worse (353).

We do an ablation study by comparing Hill's model and AE with CPAE. Recall that Hill's model lacks the reconstruction cost while AE lacks the consistency penalty. Firstly, CPAE always improves over AE. Thus, we confirm the results of the previous section on the importance of the consistency penalty. In that setting, it is more obvious why this penalty helps, as it now provides pretrained targets to the encoder. Secondly, CPAE improves over Hill on all similarity benchmarks by a large margin (+12.2 on SimLex999, +13.7 on SimLex333, +16.1 on SimVerb3500). It is sometimes slightly worse on relatedness benchmarks (−3.3 on MEN-test, −5.6 on MTurk), other times better or equal. We conclude that both terms of the CPAE objective matter.

We see identical trends when using the full Wikipedia dump. As expected, CPAE can still improve over even higher quality embeddings by roughly the same margins. The results are presented in Appendix A.4.

Remarkably, the best model among all our experiments is CPAE in Table 1 and uses only the dictionary data. This supports our hypothesis that dictionaries contain similarity-specific information.

5.7. Generalisation on unseen definitions

A model that uses definitions to produce word representations is appealing because it could be extremely data-efficient. Unlike regular distributional methods which iteratively refine their representation as occurrences accumulate, such a model could output a representation in one shot. We now evaluate CPAE in a setting where some definitions are not seen during training.

The dictionary is split into train, validation (for early stopping) and test splits. The algorithm for splitting the dictionary puts words in batches. It ensures two things: firstly, that words which share at least one definition are in the same batch, and secondly, that each word in a batch is associated with all its definitions. We can then group batches to build the

training and the test sets such that the test set does not contain synonyms of words from the other sets. We sort the batches by the number of distinct definitions they contain. We use the largest batch returned by the algorithm as the training set: it contains mostly frequent and polysemous words. The validation and the test sets, on the contrary, contain many multiword expressions, proper nouns, and rarer words. More details are given in Appendix A.2.1.

We train CPAE only on the train split of the dictionary, with randomly initialized input embeddings. Table 3 presents the same correlation coefficients as in the previous tables but also distinguishes between two subsets of the pairs: the pairs for which all the definitions were seen during training (*train*) and the pairs for which at least one word was defined in the test set (*test*). Unfortunately, there are not enough pairs of words which both appear in the test set to be able to compute significant correlations. On small-sized benchmarks, correlation coefficients are sometimes not significant so we do not report them (when p-value > 0.01).

The scores of CPAE on the test pairs are quite correlated with the ground truth: except on SimLex999 and SCWS, there is no drop in correlation coefficients between the two sets. The scores of Hill's model follow similar trends, but are lower on every benchmark so we do not report them. This shows that recurrent encoders are able to generalize and produce coherent embeddings as a function of other embeddings in one pass.

5.8. Conclusion and future work

We have focused on capturing the similarity relation. It is a challenging task which we have proposed to solve using dictionaries, as definitions seem to encode the relevant kind of information.

We have presented an alternative for learning word embeddings that uses dictionary definitions. As a definition autoencoder, our approach is self-contained, but it can alternatively be used to improve pretrained embeddings, and includes Hill's model [Hill et al., 2016b] as a special case.

In addition, our model leverages the inherent recursivity of dictionaries via a consistency penalty, which yields significant improvements over the vanilla autoencoder.

Our method outperforms dict2vec and retrofitting on similarity benchmarks by a quite large margin. Unlike dict2vec, our method can be used as a postprocessing method which does not require going through the original pretraining corpus, it has fewer hyperparameters, and it generalises to new words.

We see several directions for future work.

Firstly, more work is needed to evaluate the representations on other languages and tasks.

Secondly, solving downstream tasks requires representations for the inflected words as well. We have set aside this issue by focusing on benchmarks involving lemmas. To address it in future work, we might want to split word representations into a lexical and a morphological

part. With such a split representation, we could postprocess only the lexical component, and all the words, whether inflected or not, would benefit from this. This seems desirable for postprocessing methods in general and would make them more suitable for synthetic languages.

Thirdly, dictionary defines every sense of words, so we could produce one embedding per sense [Chen et al., 2014, Iacobacci et al., 2015]. This requires potentially complicated modifications to our model as we would need to disambiguate senses inside each definition. However, some class of words might benefit a lot from such representations, for example words that can be used as different parts of speech.

Lastly, a more speculative direction could be to study iterative constructions of the set of embeddings. As our algorithm can generalize in one shot, we could start the training with a small set of words and their definitions and iteratively broaden the vocabulary and refine the representations without retraining the model. This could be useful in discovering a set of semantic primes from which one can define all the other words [Wierzbicka, 1996].

Acknowledgements

We thank NSERC and Facebook for financial support, Calcul Canada for computational resources, Dzmitry Bahdanau and Stanisław Jastrzębski for contributing to the code on which the implementation is based, the developers of Theano [Theano Development Team, 2016], Blocks and Fuel [van Merriënboer et al., 2015] as well as Laura Ball for proofreading.

Chapter 6

Prologue to second article: Do sequence-to-sequence VAEs learn global features of sentences?

Abstract:

Autoregressive language models are powerful and relatively easy to train. However, these models are usually trained without explicit conditioning labels and do not offer easy ways to control global aspects such as sentiment or topic during generation. Bowman et al. [2016] adapted the Variational Autoencoder (*VAE*) for natural language with the sequence-to-sequence architecture and claimed that the latent vector was able to capture such global features in an unsupervised manner. We question this claim. We measure which words benefit most from the latent information by decomposing the reconstruction loss per position in the sentence. Using this method, we find that VAEs are prone to memorizing the first words and the sentence length, producing local features of limited usefulness. To alleviate this, we investigate alternative architectures based on bag-of-words assumptions and language model pretraining. These variants learn latent variables that are more global, i.e., more predictive of topic or sentiment labels. Moreover, using reconstructions, we observe that they decrease memorization: the first word and the sentence length are not recovered as accurately than with the baselines, consequently yielding more diverse reconstructions.

Bibliographical entry: Tom Bosc and Pascal Vincent. Do sequence-to-sequence VAEs learn global features of sentences? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4296–4318, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.350

6.1. Context

At the time this work was in preparation, Transformer-based auto-regressive models such as GPT2 [Radford et al., 2019] produced the best text quality. The main method to control

text generation was to condition them on a prompt – the beginning of the text that it would complete. But it was not clear how to control these models via global attributes, for example, to produce texts about a certain topic, using a positive or a negative tone.

I had worked on a VQVAE-based model [van den Oord et al., 2017] to explicitly model various correlations in texts: global correlations between words which indicate sentiment and topic, but also correlations between words denoting the same entity. It was significantly more complicated than the seq2seq VAE [Bowman et al., 2016] and did not work. In order to understand why, I tried to analyze the seq2seq-VAE [Bowman et al., 2016] in more detail. It did not work as well as I thought either, which led to this paper.

6.1.1. Modifications to the published article

Following Tim O'Donnell's comments, we have added a paragraph in B.2.2 discussing why it is less clear that the last tokens (as opposed to sentence length) are memorized.

6.2. Personal contributions

I designed the baselines, ran the experiments, and did most of the writing. Most of the code was taken from the work of Li et al. [2019], except for the baselines and the metrics. Pascal helped with the writing, the rebuttals, and provided high-level guidance throughout the project.

6.3. Contributions

- Visualization of memorization of first words in seq2seq-VAEs.
- Introduction of an automated metric, *agreement*, for evaluating controllable generation.
- Clarification of the evaluation process: the use of z vs μ, important difference between datasets.
- Introduction of simple baselines that suffer less from memorization.

Chapter 7

Do sequence-to-sequence VAEs learn global features of sentences?

7.1. Introduction

The problem of generating natural language underlies many classical NLP tasks such as translation, summarization, paraphrasing, etc. The problem is often formulated as learning a probabilistic model of sentences, then searching for probable sentences under this model. Expressive language models are typically built using neural networks [Bengio et al., 2003, Mikolov et al., 2010].

Whether based on LSTMs [Hochreiter and Schmidhuber, 1997, Sundermeyer et al., 2012] or Transformers [Vaswani et al., 2017, Radford et al., 2019], language models are mostly autoregressive: the probability of a sentence is the product of the probability of each word given the previous words. By contrast, Bowman et al. [2016] built a Variational Autoencoder (*VAE*) [Kingma and Welling, 2013, Rezende et al., 2014] out of a sequence-to-sequence architecture (*seq2seq*) [Sutskever et al., 2014]. It generates text in a two-step process: first, a latent vector is sampled from a prior distribution; then, words are sampled from the probability distribution produced by the autoregressive decoder, conditionally on the latent vector. The goal was to encourage a useful information decomposition, where latent vectors would "explicitly model holistic properties of sentences such as style, topic, and high-level syntactic features" [Bowman et al., 2016], while the more local correlations would be handled by the recurrent decoder.

In principle, such a decomposition can be the basis for many applications. For example, using a single, unannotated corpus, it could enable paraphrasing [Roy and Grangier, 2019] or style transfer [Xu et al., 2019]. For tasks requiring conditional generation such as machine translation or dialogue modeling, we could enforce a level of formality or impose a certain tone by clamping the latent vector. Moreover, latent-variable models can represent multimodal distributions. Thus, for these conditional tasks, the latent variable can be used as a source

of stochasticity to ensure more diverse translations [Pagnoni et al., 2018] or answers in a dialogue [Serban et al., 2017].

Despite its conceptual appeal, Bowman et al. [2016]'s VAE suffers from the *posterior collapse* problem: early on during training, the KL term in the VAE optimization objective goes to 0, such that the approximate posterior becomes the prior and no information is encoded in the latent variable. Free bits are a popular workaround [Kingma et al., 2016] to ensure that the KL term is above a certain level, thereby enforcing that *some* information about the input is encoded. But this information is not necessarily global. After all, posterior collapse can be solved trivially, without any learning, using encoders that copy parts of the inputs in the latent variable, yielding very local and useless features.

In Section 7.3, we show that encoders learn to partially memorize the first few words and the document lengths, as was first discovered by Kim et al. [2018]. To do so, we compare the average values of the reconstruction loss at different positions in the sentence to that of an unconditional language model. We elaborate on the negative consequences of this finding for generative models of texts. In Section 7.4, we propose three simple variants of the model and the training procedure, in order to alleviate memorization and to yield more useful global features. In Section 7.5, we empirically confirm that our variants produce more global features, i.e., features more predictive of global aspects of documents such as topic and sentiment. They do so while memorizing the first word and the sentence length less often, as shown in Section 7.6.

7.2. Model and datasets

Firstly, we describe the VAE based on the seq2seq architecture of Bowman et al. [2016]. A document, sentence or paragraph, of L words $x = (x_1, \ldots, x_L)$ is embedded in L vectors $(e_1, \ldots, e_L)$. An LSTM encoder processes these embeddings to produce hidden states:

$$h_1, \ldots, h_L = \text{LSTM}(e_1, \ldots, e_L)$$

In general, the encoder produces a vector r that represents the entire document. In the original model, this vector is the hidden state of the last word $r = h_L$, but we introduce variants later on. This representation is transformed by linear functions L_1 and L_2, yielding the variational parameters that are specific to each input document:

$$\mu = L_1 r$$
$$\sigma^2 = \exp(L_2 r)$$

These two vectors of dimension d fully determine the approximate posterior, a multivariate normal with a diagonal covariance matrix, $q_\phi(z|x) = \mathcal{N}(z|\mu, \text{diag}(\sigma^2))$, where ϕ is the set of all encoder parameters (the parameters of the LSTM, L_1 and L_2). Then, a sample z is drawn

from the approximate posterior, and the decoder, another LSTM, produces a sequence of hidden states:

$$h'_1, \ldots, h'_L = \text{LSTM}([e_{\text{BOS}}; z], [e_1; z], \ldots, e_L; z])$$

where BOS is a special token indicating the beginning of the sentence and $[\cdot;\cdot]$ denotes the concatenation of vectors. Finally, each hidden state at position i is transformed to produce a probability distribution of the word at position $i+1$:

$$p_\theta(x_{i+1}|x_{1,\ldots,i}, z) = \text{softmax}(Wh'_i + b)$$

where $\text{softmax}(v_i) = e^{v_i} / \sum_j e^{v_j}$ and θ is the set of parameters of the decoder (the parameters of the LSTM decoder, W and b). An EOS token indicating the end of the sentence is appended to every document.

For each document x, the lower-bound on the marginal log-likelihood (*ELBo*) is:

$$\begin{aligned}\text{ELBo}(x, \phi, \theta) &= -D_{\text{KL}}(q_\phi(z|x)||p(z)) + \\ &\qquad \mathbb{E}_{q_\phi}[\log p_\theta(x|z)] \\ &\leq \log p(x)\end{aligned}$$

On the entire training set $\{x^{(1)},.,x^{(N)}\}$, the objective is:

$$\arg\max_{\phi,\theta} \sum_{j=1}^{N} \text{ELBo}(x^{(j)}, \phi, \theta)$$

7.2.1. Dealing with posterior collapse

Following Alemi et al. [2018], we call the average value of the KL term the *rate*. It measures how much information is encoded on average about the datapoint x by the approximate posterior $q_\phi(z|x)$. When the rate goes to 0, the posterior is said to *collapse*, meaning that $q_\phi(z|x) \approx p(z)$ and that the latent variable z sampled to train the decoder does not contain any information about the input x.

To prevent this, we can modify the KL term to make sure it is above a target rate using a variety of techniques (see Appendix B.1.1 for a small survey). We use the free bits formulation of the δ-VAE [Razavi et al., 2019]. For a desired rate λ, the modified negative ELBo is:

$$\max(D_{\text{KL}}(q_\phi(z|x)||p(z)), \lambda) - \mathbb{E}_{q_\phi}[\log p_\theta(x|z)]$$

Seq2seq VAEs are prone to posterior collapse, so in practice, the rates obtained are very close to the target rates λ.

As observed by Alemi et al. [2018], different models or sets of hyperparameters for a given model can yield very similar values of ELBos despite reaching very different rates. Thus, for our purposes, the free bits modification is also useful to compare models with similar capacity.

7.2.2. Variants

Throughout the paper, we use variants of the original architecture and training procedure. In general, these variants use free bits objectives, but reach lower perplexities than what free bits alone allow.

Li et al. [2019]'s method is the following: pretrain an *AE*, reinitialize the weights of the decoder, train the entire model again end-to-end with the VAE objective. The sentence representation r is also the last hidden state of the LSTM encoder, so we call this method *last-PreAE*.

In the second variant, proposed by Long et al. [2019], the representation of the document r is the component-wise maximum over hidden states h_i, i.e., $r^j = \max_i h_i^j$. We call this model *max*. In later experiments, we also consider a hybrid of the two techniques, *max-PreAE*.

We chose these two baselines because they are relatively recent and outperform or perform on par with other recent methods such that cyclical learning rates [Fu et al., 2019] or aggressive training [He et al., 2019]. Moreover, the pooling encoder of Long et al. [2019] is particularly interesting: since pooling operators aggregate information over sets of vectors, they might prevent the copying of local information in the latent variable.

We make slight, beneficial modifications to these two methods. We remove KL annealing, which is not only redundant with the free bits technique but also increases the rate erratically [Pelsmaeker and Aziz, 2019]. Moreover, for Li et al. [2019]'s method, we use δ-VAE-style free bits instead of the original free bits to get rid of the unnecessary constraint that the free bits be balanced across components. For more details, see Appendix B.1. In summary, all of our experiments use δ-VAE-style free bits without KL annealing.

Finally, *AE* denotes the deterministic autoencoder trained only with the reconstruction loss.

7.2.3. Datasets

We train VAEs on four small versions of the AGNews, Amazon, Yahoo, and Yelp datasets created by Zhang et al. [2015]. Each document is written in English and consists of one or several sentences. Each document is labeled manually according to its main topic or the sentiment it expresses, and the labels are close to uniformly balanced over all the datasets. For faster training, we use smaller datasets. The characteristics of these datasets are detailed in Table 3 in the Appendix.

7.3. Encoders partially memorize the first words and sentence length

The ELBo objective trades off the KL term against the reconstruction term. To minimize the objective, it is worth increasing the KL term only if the reconstruction term is decreased by the same amount or more. With free bits, the encoder is allowed to store a fixed amount of information for free. The objective becomes to minimize the reconstruction cost using the "free storage" as efficiently as possible.

There are many solutions to this objective that are undesirable. For instance, we could program an encoder that would encode the words into the latent variable losslessly, until all the free bits are used. However, this model would not be more useful than a standard, left-to-right autoregressive models. Therefore, it is necessary to check that such useless, purely local features are not learned.

In order to visualize *what* information is stored in the latents, our method is to look at *where* gains are seen in the reconstruction loss. Since the loss is a sum over documents and positions in these documents, these gains could be concentrated: i) on certain documents, for example, on large documents or documents containing rarer words; ii) at certain positions in the sentence, for example, in the beginning or in the middle of the sentence. We investigate the latter possibility.[1]

7.3.1. Visualizing the reconstruction loss

Concretely, we compare the reconstruction loss of different models at different positions in the sentence. The baseline is a LSTM trained with a language model objective (*LSTM-LM*). It has the same size as the decoders of the autoencoder models.[2] Since the posterior collapse makes VAEs behave exactly like the *LSTM-LM*, the reconstruction losses between the VAEs and the *LSTM-LM* are directly comparable. Additionally, the deterministic *AE* gives us the reconstruction error that is reachable with a latent space constrained only by its dimension d, but not by any target rate λ (equivalent to an infinite target rate).

In Figure 1, the left-hand side plot shows the reconstruction losses of different models and different target rates λ on the Yahoo dataset. As expected, for all models, raising the target rate lowers the reconstruction cost. Remarkably, these gains are very focused around the beginning and the end of documents. For a clearer picture of the gains at the end of the sentence, we plot the *relative improvement* in reconstruction with respect to the baseline (right-hand side of Figure 1) using:

[1] PyTorch [Paszke et al., 2019] implementation available at `https://github.com/tombosc/exps-s2svae`.
[2] Only the input dimensions slightly change because in VAEs, the inputs of the decoder also include the latent vector.

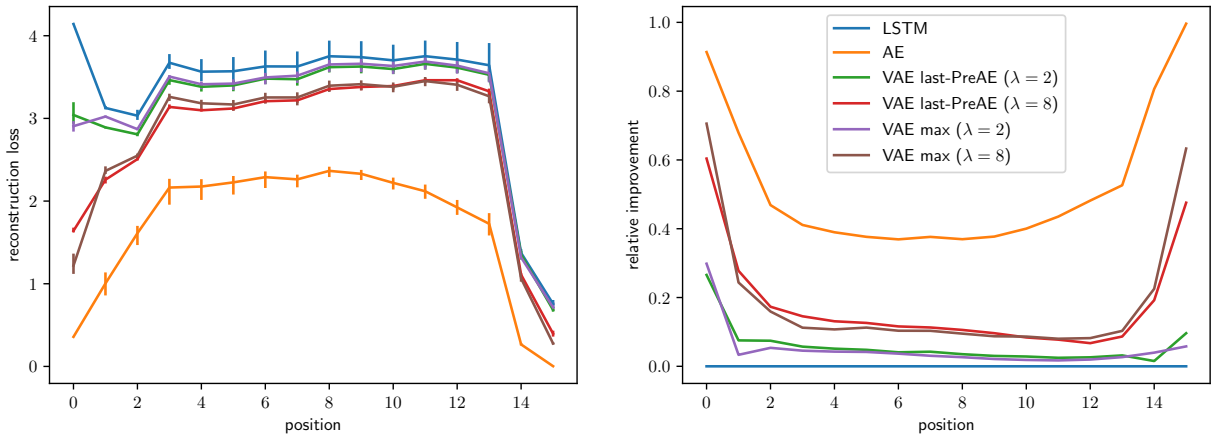

Figure 1. *Left*: Reconstruction loss on Yahoo dataset per each position in the sentence, averaged over sentences of 15 words (error bars: min, max on 3 runs); *Right*: Relative improvement compared to baseline LSTM. Seq2seq autoencoders consistently store information about the first couple of words as well as the sentence length in priority.

$$\tilde{r}(i) = \frac{\max(r_{\text{LSTM}}(i) - r(i), 0)}{r_{\text{LSTM}}(i)}$$

where $r_{\text{LSTM}}(i)$ is the loss of the LSTM.

All the models reconstruct the first couple of words and the penultimate token better than the *LSTM-LM*. On the three other datasets, there are similar peaks on relative improvements in the beginning and the end of sentences (Appendix B.2.1).

It is not obvious that a lower reconstruction at a given position corresponds to information stored about the word in that position in the latent vector. Indeed, words are not independently modeled. However, we argue that it is roughly the case because the decoder is factorised from left-to-right and because correlations between words decrease with their distance in the sentence. The argument is detailed in the Appendix B.2.2.

How much do these gains on the reconstruction loss translate to decoding the first words and the document lengths more accurately? To find out, we compare regular VAEs to fixed-encoder, ideal VAEs that encode the true label perfectly and exclusively (in other words, VAEs whose latent variable is the ground-truth label). On sentence reconstruction, we found that regular VAEs decoded the first word 2 to 5 times more often than the baselines, indicating memorization of the first word. We also found similar but less dramatic results for sentence length (see Appendix B.2.3 for details).

This phenomenon was already noticed by Kim et al. [2018], using a different method (saliency measures, see Appendix B.4.2 for details).

To sum up, compared to an unconditional *LSTM-LM*, the seq2seq VAEs incur a much lower reconstruction loss on the first tokens and towards the end of the sentence (around 50% less with $\lambda = 8$). Moreover, if the latent variable of the VAEs did encode the label perfectly and exclusively, they would reconstruct the first words or recover sentence length with much

lower accuracy than what is observed. Therefore, we conclude that seq2seq VAEs are biased towards memorizing the first few words and the sentence length.

7.3.2. The problem with memorization

One could argue that this is a superficial problem, as we can always give the model more free bits and decrease the loss in intermediary positions. However, this is not so simple because increasing capacity leads to a worse model fit, as was noted by Alemi et al. [2018]. More specifically, on text data, Prokhorov et al. [2019] noted that the coherence of samples decreases as the target rate increases. Pelsmaeker and Aziz [2019] reported similar findings, and also, that more complex priors or posteriors do not help. Therefore, given current techniques, higher rates come at the cost of worse modeling of the data and therefore, we should strive for latent-variable models that store less information, but more global information.

Secondly, for controllable generation, conditioning on memorized information is useless. When the first words are encoded in the latent variable, the factorization of the VAE becomes the same as that of the usual autoregressive models, which are naturally able to continue a given beginning of the sentence (a "prompt"). Similarly, document length is easily controlled by stopping the sampling after producing the desired number of words.[3] Finally, even for semi-supervised learning, a classifier that would only use the first few words and the sentence length would be suboptimal.

If these arguments are correct, it is doubtful that common seq2seq VAE architectures and training procedures in the low-capacity regime would learn useful representations. This is precisely the third problem: most of the KL values reported in the literature are low.[4] Therefore, it is not clear whether the reported gains in performance (however measured) are significant, and if they are, what exactly cause these gains.

7.4. Improving existing models

What architectures could avoid learning to memorize? We investigate simple variants and for a more thorough comparison with existing models, we refer to Appendix B.4.1.

Our first variant uses a simple bag-of-words (BoW) encoder in place of the LSTM encoder. The sentence representation is $r^j = \max_i e_i^j$, where the exponents denote components, and the indices denote positions in the sentence. We call it *BoW-max-LSTM*. It is similar to the max-pooling model of Long et al. [2019] except that the maximum is taken over embeddings

[3]Or by explicitly conditioning on the sentence length. It can be useful for unsupervised summarization [Schumann, 2018], in flow-based approaches [Ziegler and Rush, 2019], or more broadly for the decoder to plan sentence construction.

[4]Most papers do not report the log base (1 bit is $\ln(2) \approx 0.693$ nats). Here are some reported rates of the best models: Bowman et al. [2015]: 2.0 (PTB) ; Long et al. [2019]: 3.7 (Yahoo), 3.1 (Yelp); Li et al. [2019]: 15.02 (Yahoo), 8.15 (PTB); He et al. [2019]: 5.6 (Yahoo), 3.4 (Yelp); Fu et al. [2019]: 1.955 (PTB), ...

rather than LSTM hidden states. As Long et al. [2019] reported, the max-pooling operator is better than the average operator, both when the encoder is a LSTM and *BoW* (possibly because the maximum introduces a non-linearity). Therefore, we use the maximum operator. A priori, we think that since word order is not provided to the encoder, the encoder should be unable to memorize the first words.

For our second variant we use a unigram decoder (*Uni*) in place of an LSTM decoder. It produces a single output probability distribution for all positions in the sentence i, conditioned only on the latent variable z. This distribution is obtained by applying a one-hidden layer MLP followed by softmax to the latent vector: $p_\theta(x_i|z) = \text{softmax}(W_2\text{ReLU}(W_1 z) + b)$, where $\text{ReLU}(x) = \max(x,0)$ [Nair and Hinton, 2010a]. We hope that the encoder will learn representations that do not focus on the first words, because the decoder should not need this particular information. We can use any encoder in combination of this decoder and if we use a *BoW* encoder, we obtain the NVDM model of Miao et al. [2016].

Both the *BoW* encoders and *Uni* decoders variants might benefit from the *PreAE* pretraining technique, but we leave this for future work.

Lastly, the pretrained LM (*PreLM*) variant is obtained in two training steps. First, we pretrain a *LSTM-LM*. Then, it is used as an encoder with fixed weights. We use average pooling over the hidden states to get a sentence representation, i.e., $r = \frac{1}{L}\sum_{i=1}^{L} h_i$, and learn the transformations L_1 and L_2 that compute the variational parameters. Initially, we tried to use max-pooling but the training was extremely unstable. The LM objective requires the hidden state to capture both close correlations between words but also more global information to predict long-distance correlations. The hope is that this global information can be retrieved via pooling and encoded in the variational parameters. The *PreLM* variant is nothing more than the use of a pretrained LM as a feature extractor [Peters et al., 2018]. While Yang et al. [2017] and Kim et al. [2018] both consider the use of pretrained LMs as encoders, the weights are not frozen such that it is hard to disentangle the impact of pretraining from subsequent training. In contrast, we freeze the weights so that the effect of pretraining can not be overridden. To isolate the effect of this training procedure independently of the architecture, we keep the same LSTM instead of using more powerful architectures such as Transformers.

7.5. Semi-supervised learning evaluation

We turn to the semi-supervised learning (*SSL*) setting to compare the learned representations of our variants. For the purpose of controllable text generation, we assume that the global information that is desirable to capture is the topic or sentiment. There are two training phases: first, an unsupervised pretraining phase where VAEs are trained; second, a supervised learning phase where classifiers are trained to predict ground-truth labels given the latent vectors encoded with the encoders of the VAEs. This is essentially the same setup as *M1* from

Kingma et al. [2014].[5] The small and large data-regimes give us complementary information: with many labels and complex classifiers, we quantify *how much* of the information pertaining to the labels is encoded; with few labels and simple classifiers, *how accessible* the information is.

For each dataset, we subsample $g = 5$ balanced labeled datasets for each different data-regimes, containing 5, 50, 500, and 5000 examples per class. These labeled datasets are used for training and validating during the supervised learning phase.[6] Each model is trained with $s = 3$ seeds. The performance of the classifiers are measured by the macro F1-score on the entire test sets.

To select hyperparameters on each subsample, we use repeated stratified K-fold cross-validation [Moss et al., 2018] as detailed in the Appendix B.5.1. We obtain the test set F1-scores F_{ij}, where i is the subsample seed and j is the parameter initialisation seed, and report $\bar{F}_{..}$, the average F1-score over i and j. We note $\bar{F}_{.j}$ the empirical average F1-score for a given parameter initialisation j and decompose the variance into:

- $\sigma_{\text{init}} = (\frac{1}{s-1}\sum_{j=1}^{s} g(\bar{F}_{.j} - \bar{F}_{..})^2)^{\frac{1}{2}}$, which quantifies the variability due to the initialisation of the model,
- $\sigma = (\frac{1}{g}\sum_{i=1}^{g}\frac{1}{s-1}\sum_{j=1}^{s}(F_{ij} - \bar{F}_{.j})^2)^{\frac{1}{2}}$, which quantifies the remaining variability.

In the context of ANOVA with a linear model and a single factor, these quantities are the square roots of MS_T and MS_E (see Appendix B.5.2).

Finally, we also add a data-regime where the entire labeled training set is used in the supervised learning phase. In this setting, we use more expressive one-hidden-layer MLP classifiers, with early stopping on the validation set. Thus, we can check that our conclusions in the large data-regime do not depend on the model selection procedure and the choice of the classifier.

For each class of model, we perform a grid search over target rates and latent vector sizes. We search for target rates λ in {2,8}: large enough to capture label information but small enough to avoid underfitting, as explained above. The size of latent vectors d are chosen in {4,16}. They should be small enough for extremely low-data regimes. For instance, on Yelp, the smallest data regime (5 per class) uses only 8 examples to train the classifier and 2 to do cross-validation. A thorough explanation is presented in Appendix B.3, along with the values of hyperparameters held constant.

What representation should be used as inputs to the classifiers? Kingma et al. [2016] use samples from the approximate posterior $q_\phi(z|x) = \mathcal{N}(z|\mu, \text{diag}(\sigma^2))$, but in the NLP

[5]We could integrate the labels into the generative model as a random variable that is either observed or missing to obtain better results [Kingma et al., 2014]. Still, our goal is to study the inductive bias of the seq2seq VAE *as an unsupervised learning method*, so we do not train the encoder using the labels.

[6]It is especially important to use several subsamples in the low data-regimes where subsamples containing unrepresentative texts or noisy labels are not unlikely.

	Enc.	r	Dec.	Pre.	5 F1	All σ	σ_{init}
AGNews	LSTM	last	LSTM	AE	65.8	83.4	$^{3.3}_{3.3}$ $^{-}_{0.3}$
	LSTM	max	LSTM	AE	55.7	83.3	$^{4.5}_{18.7}$ $^{-}_{0.4}$
	BoW	max	LSTM	-	72.7	83.1	$^{2.0}_{5.9}$ $^{-}_{0.3}$
	LSTM	max	Uni	-	71.6	83.9	$^{5.5}_{0.1}$ $^{-}_{0.3}$
	LSTM	last	Uni	-	54.8	59.3	$^{5.2}_{57.1}$ $^{-}_{40.9}$
	BoW	max	Uni	-	71.8	83.1	$^{4.5}_{1.8}$ $^{-}_{0.5}$
	LSTM	avg	LSTM	LM	70.8	83.5	$^{4.8}_{4.3}$ $^{-}_{0.1}$
Amazon	LSTM	last	LSTM	AE	20.0	28.1	$^{2.2}_{0.9}$ $^{-}_{1.0}$
	LSTM	max	LSTM	AE	22.3	34.0	$^{2.6}_{0.7}$ $^{-}_{1.6}$
	BoW	max	LSTM	-	21.0	38.9	$^{2.6}_{1.1}$ $^{-}_{0.7}$
	LSTM	max	Uni	-	21.8	38.2	$^{3.1}_{1.6}$ $^{-}_{0.5}$
	LSTM	last	Uni	-	24.0	36.8	$^{3.0}_{1.0}$ $^{-}_{0.9}$
	BoW	max	Uni	-	25.4	37.9	$^{3.2}_{0.2}$ $^{-}_{0.2}$
	LSTM	avg	LSTM	LM	21.8	40.0	$^{3.8}_{0.6}$ $^{-}_{0.4}$
Yahoo	LSTM	last	LSTM	AE	20.7	37.2	$^{0.7}_{0.5}$ $^{-}_{0.7}$
	LSTM	max	LSTM	AE	20.8	36.6	$^{1.3}_{2.3}$ $^{-}_{0.7}$
	BoW	max	LSTM	-	23.4	42.6	$^{2.1}_{2.9}$ $^{-}_{0.2}$
	LSTM	max	Uni	-	24.9	38.9	$^{1.3}_{2.2}$ $^{-}_{1.7}$
	LSTM	last	Uni	-	24.5	37.1	$^{3.8}_{1.7}$ $^{-}_{2.3}$
	BoW	max	Uni	-	24.1	40.1	$^{2.9}_{2.7}$ $^{-}_{0.7}$
	LSTM	avg	LSTM	LM	21.9	41.7	$^{2.3}_{1.3}$ $^{-}_{0.3}$
Yelp	LSTM	last	LSTM	AE	59.3	67.9	$^{5.4}_{2.9}$ $^{-}_{0.1}$
	LSTM	max	LSTM	AE	59.9	84.1	$^{10.4}_{7.9}$ $^{-}_{0.7}$
	BoW	max	LSTM	-	67.1	85.0	$^{10.1}_{15.7}$ $^{-}_{0.2}$
	LSTM	max	Uni	-	62.3	83.1	$^{4.6}_{3.8}$ $^{-}_{0.5}$
	LSTM	last	Uni	-	65.0	81.6	$^{8.0}_{4.4}$ $^{-}_{0.5}$
	BoW	max	Uni	-	59.9	83.3	$^{7.2}_{3.7}$ $^{-}_{0.4}$
	LSTM	avg	LSTM	LM	63.6	84.4	$^{7.4}_{5.4}$ $^{-}_{0.5}$

Table 1. Using *BoW* encoders, *Uni* decoders or *PreLM* pretraining, the learned representations are more predictive of the labels (sentiment or topic).

literature, most evaluations focus on μ without mention or justification. To evaluate the VAE as a generative model, we claim that only noisy samples z should be used. In fact, using a model with a rate close to 0 on Yelp, we can recover the label with a high F1-score of 81.5% by using μ, whereas, as expected, noisy samples z do not do better than random (50%). The information contained in μ is misleading because it is not transmitted to the decoder and not used directly during generation. Therefore, we use samples z (cf. Appendix B.5.3 for details).

7.5.1. Results

Table 1 contains the results of the SSL experiments in the smallest and largest data-regimes. The results for intermediary data-regimes as well as for baseline models without pretraining, which underperform, are presented in the Appendix, Table 6. The proposed variants are either on par or improve significantly over the baselines. In the large data-regime, *BoW-max-LSTM* and *LSTM-avg-LSTM-PreLM* perform best on average while *LSTM-last-Uni* performs the worst and suffers from unstable training on AGNews. In the small data-regime, the picture is less clear because there is more variance.

On AGNews and Yelp, in the large data-regime, our variants do not seem to improve over the baselines. However, on Amazon and Yahoo, in the large data-regime, the variants seem to improve by 5 in F1-score. Why do the gains vary so widely depending on the datasets? We posit that, on some datasets, the first words are enough to predict the labels correctly. We train bag-of-words classifiers [7] using either i) only the first three words or ii) all the words as features on the entire datasets. If the *three-words* classifiers are as good as the *all-words* classifiers, we expect that the original VAE variants will perform well: in that case, encoding information about the first words is not harmful, it could be a rather useful inductive bias. Conversely, if the first three words are not predictive of the label, the original VAEs will perform badly.

As reported in the Appendix, Table 7, on AGNews and Yelp, classifiers trained on the first three words have a performance somewhat close to the classifier trained on all the words, reaching 80.8% and 85.4% of its scores respectively. For instance, on AGNews, the first words are often nouns that directly gives the topic of the news item: country names for the politics category, firm names for the technology category, athlete or team names for the sports category, etc. On the two other datasets, the performance decays a lot if we only use the first three words: *three-words* F1-scores make up for 60.7% and 30.3% of *all-words* F1-scores on Amazon and Yahoo. This explains why the original VAE can perform on par or slightly better than our variants on certain datasets for which the first words are very predictive of the labels. This also proves that using several datasets is necessary to draw robust conclusions.

Despite similar asymptotic performance on AGNews and Yelp, our variants clearly improve over the baselines in the small data-regime, which suggests that the encoded information is quantitatively different. This is confirmed in the next section.

It might be surprising that *LSTM-max-LSTM* models are inferior to *BoW-max-LSTM* models. In Appendix B.5.4, we show that with recurrent encoders, some components of the hidden states are consistently maximized at certain early positions in the sentence. This

[7] fastText classifiers [Joulin et al., 2017] with embedding dimension of 200 and the default parameters.

Enc.	r	Pre.	Agree.	1st (%)	Len (%)	≈PPL
LSTM	last	AE	80.2±1.0	29.6±1.1	3.6±0.1	**34.8±0.4**
LSTM	max	AE	79.5±0.9	31.7±1.1	3.7±0.5	**34.7±0.4**
BoW	max	-	78.0±1.3	18.9±1.2	**2.7±0.3**	36.1±0.6
BoW	max	Uni	81.3±0.1	**13.9±0.3**	3.1±0.1	36.3±0.7
LSTM	max	Uni	**82.0±0.4**	**13.9±0.2**	3.3±0.4	36.0±0.4
LSTM	avg	LM	79.2±0.4	22.2±0.8	3.2±0.2	**35.0±0.3**
LSTM	last	AE	24.5±0.4	42.4±2.3	13.0±1.6	**44.5±0.2**
LSTM	max	AE	30.8±1.1	41.7±0.8	11.5±1.0	**44.4±0.3**
BoW	max	-	34.2±0.5	33.3±0.7	**9.9±0.7**	45.3±0.5
BoW	max	Uni	33.3±0.4	**21.5±0.3**	11.8±0.5	45.3±0.4
LSTM	max	Uni	34.1±0.5	22.1±0.1	11.7±0.6	45.4±0.6
LSTM	avg	LM	**35.8±0.4**	38.3±0.9	11.5±1.0	**44.2±0.4**
LSTM	last	AE	23.8±0.2	56.6±1.0	17.1±1.1	**48.8±0.2**
LSTM	max	AE	22.9±0.8	58.7±1.7	18.4±0.8	**48.6±0.1**
BoW	max	-	**26.9±0.5**	49.3±1.2	11.8±0.3	49.7±0.4
BoW	max	Uni	**26.8±0.6**	**37.6±0.9**	**10.6±0.4**	49.8±0.1
LSTM	max	Uni	**27.1±1.0**	**37.7±1.6**	**11.0±0.3**	50.0±0.4
LSTM	avg	LM	**26.7±0.2**	51.9±0.5	16.7±1.8	**48.5±0.1**
LSTM	last	AE	**81.7±1.3**	53.0±0.5	33.7±1.7	**31.7±0.3**
LSTM	max	AE	81.3±0.7	52.4±0.5	29.5±2.5	**31.8±0.1**
BoW	max	-	**82.2±0.5**	36.4±0.3	22.4±0.5	32.3±0.4
BoW	max	Uni	80.4±0.4	**30.6±0.5**	**15.4±0.4**	32.8±0.1
LSTM	max	Uni	80.9±0.4	32.0±0.4	17.2±0.7	33.1±0.3
LSTM	avg	LM	**82.3±0.7**	47.7±0.4	24.1±0.4	**31.9±0.2**

Table 2. Our variants reconstruct inputs with higher agreement, less memorization of the 1st words and lengths and a negligible loss in likelihood. Best score and scores within one standard deviation are bolded.

explains why the power of LSTMs can be undesirable, and why the simpler *BoW* encoders perform better.

7.6. Text generation evaluation

How do these different variants perform during generation? We expect that the SSL classification performances would correlate with the abilities of the decoders to reconstruct documents that exhibit a similar global aspect than the encoded documents.

To measure the agreement in label between the source document and its reconstruction, we adapt the evaluation procedure used by Ficler and Goldberg [2017] so that no human annotators or heuristics are required (see Appendix B.4.2). First, a classifier is trained to predict the label on the source dataset. Then, for each model, we encode the documents,

reconstruct them, and classify these reconstructions using the classifier. The *agreement* is the F1-scores between the original labels and the labels given by the classifiers on the generated samples.

To quantify memorization, we measure the reconstruction accuracy of the first word and the ratio of identical sentence length between sources and reconstructions. Finally, to verify that our bag-of-words assumptions do not hurt the overall fit to the data, we estimate the negative log-likelihood via the importance-weighted lower bound [Burda et al., 2015] (500 samples) to compute an approximate perplexity per word ($\approx PPL$).

We use two decoding schemes: beam search with a beam of size 5 and greedy decoding. We fix $\lambda = 8$, $d = 16$ on all models, with three seeds. For the *Uni* decoder, we drop *LSTM-last-Uni* which underperformed by a large margin in the SSL setting, and for the other *Uni* models, we freeze the encoder, L_1 and L_2 and train a new recurrent decoder using the reconstruction loss only. Essentially, the *Uni* decoder is an *auxiliary decoder*, as described by De Fauw et al. [2019] (see Appendix B.4.1 for details) and we denote this technique by *PreUni*.

Table 2 show the results for beam search decoding.[8] There is a close correspondence between agreement and performance on the SSL tasks in the large data-regime. Our variants have a higher agreement than the baselines, especially on Amazon and Yahoo datasets for which the memorization of the first words is especially harmful.

The baselines reconstruct the first words with very high accuracy (more than 50% of the time on Yahoo and Yelp) while our variants mitigate this memorization. For instance, *PreUni* models recover the first word around 2 or 1.5 times less often.

Let us focus on AGNews and Yelp, where the first words are very predictive of the labels. Both baselines and variants have roughly similarly high agreement. However, our variants produce more diverse beginnings, while still managing to reproduce the topic or sentiment of the original document. On the other hand, the reconstructions of the baselines exhibit the same labels as the sources mostly as a side-effect of starting with the same words. This also explains that in the SSL setting, despite similar performances asymptotically, our variants were much more efficient using five examples per class. Memorization of the first words does not abstract away from the particular words and therefore, the amount of data required to learn a good classifier will be high, compared to a model which truly *infer* unobserved characteristics of documents.

Both *BoW* encoders and *Uni* decoders lower memorization, so bag-of-words assumptions are efficient for dealing with the memorization problem. Still, *BoW-Max* and *LSTM-Max* with *PreUni* pretraining yield very close performance despite having a different encoder, showing that the decoder has a far greater influence than the encoder. This is consistent with McCoy et al. [2019]'s findings (see Section B.4.2 in Appendix for details).

[8]Similar results were obtained using greedy decoding, albeit sometimes consistently shifted.

Finally, there seems to be a trade-off between the global character of the latent information and the fit of the model to the data. *BoW* and *Uni* variants have perplexity roughly one unit above the baselines, a significant but small difference.

In Appendix B.6, we perform a qualitative analysis of reconstruction samples to illustrate these conclusions. It also sheds light on the inherent difficulty of the Yahoo dataset.

To recapitulate, the bag-of-words assumptions decrease the memorization of the first word and of the sentence length in the latent variable while increasing the agreement between the labels of the source and of the reconstruction. This is achieved at the cost of a small increase in perplexity.

7.7. Conclusion

Eliminating posterior collapse is necessary to get useful VAE models, but not sufficient. Although recent incarnations of the seq2seq VAE fix the posterior collapse, they partially memorize the first few words and the document lengths. Depending on the data, these local features are sometimes not very correlated with global aspects like topic or sentiment. Therefore, they are of limited use for controllable and diverse text generation.

To learn to infer more global features, we explored alternative architectures based on bag-of-word assumptions on the encoder or decoder side, as well as a pretraining procedure. These variants are all effective, in particular, the unigram decoder used as an auxiliary decoder [De Fauw et al., 2019]. The latent variable is more predictive of global features and memorisation of the first words and sentence length is decreased. Thus, these models are more suitable for diverse and controllable generation.

Methodologically, we introduced a simple way to examine the content of latent variables by looking at the reconstruction loss per position. We also presented a reliable way to perform semi-supervised learning experiments to analyze the content of the variable, free of the problems that one can find in past work (incorrect model selection for small data-regimes, use of samples instead of variational parameters as inputs). We showed that there are particularly difficult datasets for which the first words are not very predictive of their labels, and therefore, these datasets should be systematically used in evaluations. Moreover, the agreement metric is another complementary evaluation that is automatic and focused on generation. We hope that these methods will see widespread adoption for measuring progress more reliably.

A promising research direction is to investigate the root cause behind memorization. A simple reason for the memorization of the first few words could be that, in the beginning of training, the reconstruction loss is higher on these words (see *LSTM-LM* in Figures 1, 1, 2, 3). These early errors should therefore account for a proportionally large part of the gradients and pressure the encoder to store information about the first words. If that is correct, the left-to-right factorization of the decoder could be at fault, which would explain the successes

of the unigram decoders. More powerful decoders with alternative factorizations could avoid this issue, for example, non-autoregressive Transformers [Gu et al., 2017] or Transformers with flexible word orders [Gu et al., 2019].

VAEs operate on uncorrupted inputs and learn a corruption process in the latent space. In contrast, models in the BERT family [Devlin et al., 2018] are given corrupted inputs and are penalized only on these corrupted inputs, thereby avoid memorization altogether. Therefore, another research avenue would be to blend the two frameworks [Im et al., 2017].

Acknowledgements

We thank NSERC for financial support, Calcul Canada for computational resources and Siva Reddy, Loren Lugosch, Makesh Narasimhan and Arjun Akula for their comments on the draft, as well as the reviewers and the meta-reviewer.

Chapter 8

Prologue to third article: The Emergence of Argument Structure in Artificial Languages

Abstract: Computational approaches to the study of language emergence can help us understand how natural languages are shaped by cognitive and sociocultural factors. Previous works focused on tasks where agents *refer* to a single entity. In contrast, we study how agents *predicate*, that is, how they express that some relation holds between several entities. We introduce a setup where agents talk about a variable number of entities that can be partially observed by the listener. In the presence of a least-effort pressure, they tend to discuss only entities that are not observed by the listener. Thus we can obtain artificial phrases that denote a single entity, as well as artificial sentences that denote several entities. In natural languages, if we ignore the verb, phrases are usually concatenated, either in a specific order, or by adding case markers to form sentences. Our setup allows us to quantify how much this holds in emergent languages using a metric we call *concatenability*. We also measure *transitivity*, which quantifies the importance of word order. We demonstrate the usefulness of this new setup and metrics for studying factors that influence argument structure. We compare agents having access to input representations structured into pre-segmented objects with properties, versus unstructured representations. Our results indicate that the awareness of object structure yields a more natural sentence organisation.

8.1. Context

This work grew out of interrogations about disentanglement, compositionality and language emergence. In the generative modelling literature, some researchers focus on learning *disentangled* representations. In the works of Bengio et al. [2013] and Locatello et al. [2019], the true generating process consists in Z being sampled, then X given Z. A latent-variable model such as a VAE or a β-VAE [Higgins et al., 2016] is trained to model X, and produces a

latent vector Z'. The vector Z' is *disentangled* when it is essentially Z, up to a permutation and a rescaling of each component. In this sense, disentanglement is a similar concept to identifiability in statistics.[1] Using a dataset of colored shapes, Locatello et al. [2019] study the conditions under which a component Z'_i encodes the color and only the color, where another component Z'_j encodes the shape and only the shape, etc.

In parallel, in the language emergence community, autoencoders are used as models of human communication [Lazaridou and Baroni, 2020]. The encoder is seen as a speaker, observing an input and producing a message. The decoder plays the role of a listener, observing the message and trying to reconstruct the input of the speaker. Researchers also often work with colored shapes. They study conditions under which agents communicated about these shapes by using a word only for color, and a word only for the shape, which they define as a *compositional* language [Kottur et al., 2017, Lazaridou et al., 2018].

The tasks are formally almost identical, except that the learned representations are slightly different: continuous vectors or sequences of discrete symbols. It seems like a strange duplication of efforts. However, language emerged in the human species long after vision had developed in their ancestors [Tomasello, 2010]. Thus, we propose that language emergence research should assume access to the high-level representations that are prelinguistic, and focus on how these representations are communicated via sequences of symbols.

Yet, using disentangled representations as inputs is not enough to obtain compositional languages [Kottur et al., 2017]. A first problem might be that once these representations are collapsed into a single vector, the agents can perceive the object as a whole again. A second problem is that compositionality has a function, and for it to emerge, it needs to improve task performance.

Instead of thinking of reference problems and how the adjectives come to be, we focus on predication – how agents communicate about several entities that are in a relationship. In this case, the least-effort pressure is a very simple explanation for the emergence of compositionality. A proposal for the reference case was made in a follow-up workshop paper [Bosc, 2022].

8.2. Personal contributions

I came up with the new task and the metrics. I wrote some code, heavily relying on what is already provided in the EGG framework [Kharitonov et al., 2021] and ran the experiments. I did most of the writing. Pascal helped with the clarity of the writing of the paper and of the rebuttals. He provided high-level guidance throughout the project.

[1] There is no consensus about disentanglement. Higgins et al. [2018] provide a definition in terms of group theory, where probability and statistics are absent.

8.3. Contributions

- We introduced the new task of predication under partial observability and least-effort pressure. We argued that it is a realistic task that could explain argument structure of most natural languages.
- We introduced and motivated various metrics to quantify how "natural" the emergent languages are (concatenability and transitivity metrics).
- We identified the problem of segmentation, pervasive in language emergence papers, and showed how we can sidestep it.
- We provided evidence that appropriate representations and mechanisms help emergent languages look more "natural".

Chapter 9

The Emergence of Argument Structure in Artificial Languages

How do languages emerge and evolve? Zipf [1949] viewed language as the result of an optimisation procedure balancing information transmission maximisation and effort minimisation. This view is amenable to formalisation and simulation. An early example is Hurford [1989]'s comparison of language acquisition strategies, assuming that communication success gives an evolutionary advantage. More generally, subsequent research uses optimisation procedures and evolutionary mechanisms to create and study artificial languages [Steels, 1997, Lazaridou and Baroni, 2020].

Such approaches are mainly used with two objectives in mind: firstly, to improve natural language processing methods; secondly, to help us understand the roles of cognitive and sociocultural factors on the shape of languages, such as our drive to cooperate, pragmatic reasoning and imitation [Tomasello, 2010].

In the deep learning era, language emergence researchers have focused on the *referential* function of language, i.e. how agents communicate about single objects, using artificial noun phrases equivalent to "blue triangle" or "red circle" [Lazaridou et al., 2017, Kottur et al., 2017]. In contrast, we propose to study the *predication* function of language, i.e. the expression of relations between entities (*events*). How do artificial agents express events such as "the blue triangle is above the red circle"?

We introduce an experimental setup for studying predication. The speaker communicates about an event involving a variable number of entities that are in a certain relation. Then, the listener tries to reconstruct this event. To simplify, the relation is observed by both agents.

Crucially, the listener is given a partial observation of the event, ranging from nothing to all but one entity. In the presence of shared context, it is unnecessary for the speaker to communicate the whole event, and a least-effort penalty encourages parsimony. Thus we

obtain utterances that refer to single entities in isolation, like phrases, and utterances about several entities, like sentences.

Using these artificial phrases and sentences, we can compute various metrics to quantify compositionality [Szabó, 2020] at the sentence level. A simple sentence typically contains a few phrases which refer to entities. These phrases can generally be understood in isolation, a property sometimes called *context-independence* Bogin et al. [2019]. Moreover, the sentence is the concatenation of these phrases along with the verb. Correspondingly, we introduce *concatenability* metrics that should be large for natural languages. Furthermore, we propose *transitivity* metrics to quantify the importance of word order. A high-level overview of our setup is shown in Figure 1.

This setup enables us to analyze artificial languages without segmenting messages into constituents. Segmentation introduces another layer of complexity, to the extent that in practice, it is not done at all: it is implicitly assumed that each symbol is independently meaningful. However, this assumption is flawed, because if letters or phonemes are assumed to bear meaning, no natural language is compositional.

Previous works have highlighted the influence of input representations and architectures for language emergence. Inappropriate representations completely hinder evolution of a non-trivial language with more than 2 words [Lazaridou et al., 2017] or prevents agents from solving the task altogether [Guo et al., 2019]. This suggests that specific inductive biases are still lacking for artificial agents to develop languages like ours.

We posit that the perception of objects as wholes with properties is an important inductive bias. To be able to produce sentences containing referential phrases, it seems that agents need to be able to attend to the referents of these phrases reliably, to conceive of them as bounded objects with intrinsic properties in the first place.

We demonstrate the usefulness of our setup and our metrics for testing this hypothesis. We implement an object-centric inductive bias using attention [Bahdanau et al., 2014] over representations of objects. We compare it to an architecture which disregards the structure of the input, considering it merely a large unstructured feature vector. The object-centric architecture yields more natural languages – they are more concatenable. Furthermore, word order matters more with this architecture than for the baseline. These results are corroborated by our quantitative analysis and measures of generalisation outside of the training data.

Our contributions are two-fold. Firstly, on the methodological front, we propose and motivate a novel task and two new metrics. This task not only explains the emergence of compositionality from a functional perspective, but also enables us to easily analyze the learned language, avoiding the problem of segmentation.

Secondly, we provide evidence that when representations reflect the perception of objects as wholes with properties, emergent languages are more natural than when they do not. With

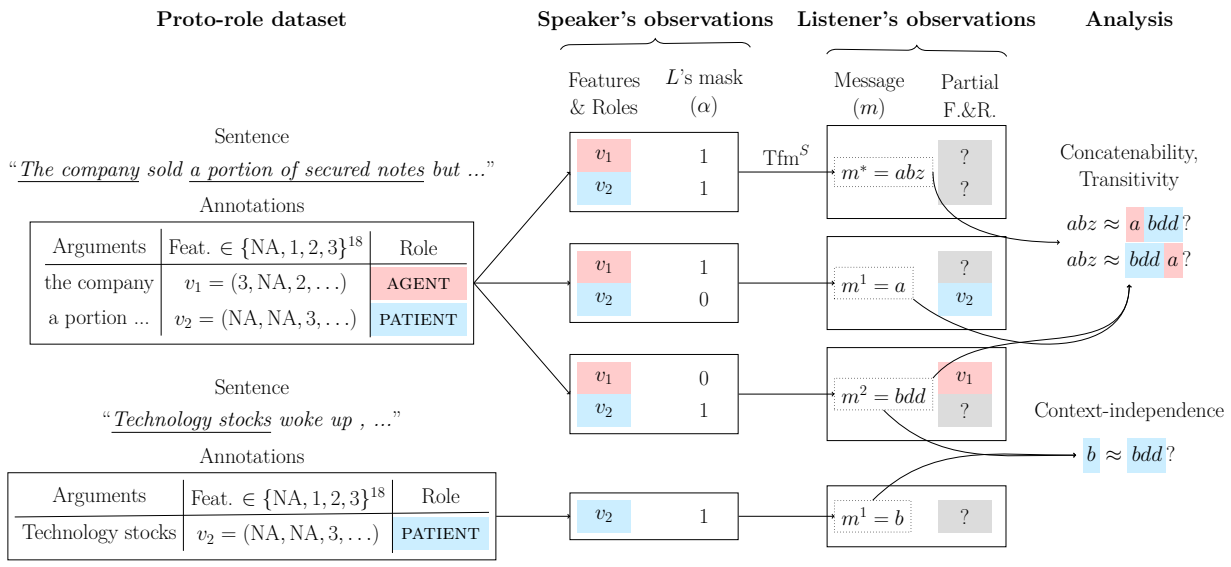

Figure 1. Overview of experimental setup. (From left to right) Proto-role dataset contains annotations (18 features and a role) for each argument and a relation (SELL.01 and WAKE.02 respectively, observed by both speaker S & and listener L). **Preprocessing:** From the 1st annotation, 3 datapoints are created, where the number of entities observed by L varies (see *L's mask* and *Partial F.&R.* columns). The 2nd annotation contains a single object so a single datapoint is created. **Training:** S produces a message. L reads it, and the pair of agents S, L is jointly trained to minimize the reconstruction error and the length of the message. As a result of the objective, S only talks about the entities not observed by L. **Analysis:** Informally, *concatenability* measures how concatenation of messages $m^{12} = m^1 \oplus m^2$ and/or $m^{21} = m^2 \oplus m^1$ are interchangeable with the actually sent message m^*; *transitivity* measures how much one order is preferred compared to the other across the dataset (cf. Sections 9.5, 9.6).

this finding we hope to foster the use of more cognitively plausible input representations for explaining language emergence.

9.1. Task

We design a task for studying how artificial agents predicate. It is an instance of a reconstruction task [Lazaridou et al., 2017], where one agent, the *speaker*, observes an input and produces a message – a sequence of symbols. The message is then read by another agent, the *listener*, who tries to reconstruct the input observed by the speaker.

We train several pairs of agents and study the messages produced by the speakers. This training procedure models language evolution and language acquisition at once, unlike frameworks like iterated learning [Kirby and Hurford, 2002].

The main novelty of our task is that agents are trained to communicate about a variable number of entities. In this section, we explain how the inputs of the agents are obtained by

preprocessing the proto-role dataset [Reisinger et al., 2015]. Then, we argue that our task is realistic, yet simple enough to permit an easy analysis of the messages.

9.1.1. The proto-role dataset

The data that are fed to agents are based on the proto-role dataset built by Reisinger et al. [2015]. This dataset was created to evaluate Dowty [1991]'s linking theory, a theory which predicts how verb-specific roles are mapped to grammatical relations in English.

To illustrate the annotation scheme, we use the example from Figure 1, "the company sold a portion of secured notes".

Firstly, a relation is extracted. Here, the verb "sold" corresponds to the PropBank [Kingsbury and Palmer, 2002] label SELL.01, which identifies the verb and its particular sense.

There are $n_{obj} = 2$ arguments of the verb, "the company" and "a portion of secured notes". Each of these argument is annotated with $n_{feat} = 18$ features indicating various properties of the referred entity. For instance, the first feature indicates whether the entity caused the event to happen, the second feature whether the entity chose to be involved in the event, etc. [Reisinger et al., 2015]. In this work, the meaning of these features is irrelevant. These features are encoded on a Likert scale from 1 and 5 or take a *non-applicable* (*NA*) value. Since the description of each entity is a small feature vector, many different noun phrases correspond to the same feature vector. Thus "Technology stocks" and "a portion of secured notes" in Figure 1 denote the same entity.

Moreover, each argument is also assigned one of six mutually exclusive classical θ-roles. In the example, the arguments respectively have the θ-roles AGENT and PATIENT.

We define an *event* as i) a relation and ii) a set of pairs of feature vectors and θ-roles.

9.1.2. Task description

For each event in the proto-role dataset, we gather the relation, and for each entity, their 18 features and their role. The features are rescaled from $\{1, 2,3,4, 5\}$ to $\{1, 2, 3\}$, and we only retain the arguments in the 3 most frequent θ-roles (AGENT, PATIENT and a MISC category containing instruments, benefactives, attributes).

The speaker observes the following quantities:

The tensors I^S, r^S and α are indexed by an integer between 1 and n_{obj}, so they represent a set E^S of n_{obj} triplets where each triplet (I_i^S, r_i^S, α_i) characterizes the i-th entity.

The i-th entity is said to be *hidden* iff $\alpha_i = 1$. Hidden entities are not observed by the listener, and the mask α indicates this to the speaker. Since the listener tries to reconstruct the inputs of the speaker, the mask essentially flags the entities that the speaker should communicate about. Thus, the listener observes:

Here, $u[v]$ denotes the tensor obtained by restricting u to the rows i such that $v_i = 1$.

The message space $\mathcal{M}$ is defined as follows. Let $\mathcal{V} = \{1, \ldots, n_V, \text{eos}\}$ be the vocabulary containing n_V symbols, plus an *end-of-sentence* (eos) token. Let n_L be the maximum message length (here, set to $n_L = 8$). $\mathcal{M}$ contains all the sequences of elements of $\mathcal{V}$ with at most length n_L and ending with eos.

A datapoint is valid if $\sum_{i=1}^{n_{obj}} \alpha_i \geq 1$, i.e. at least one object is hidden and some information needs to be conveyed. From each event, we add as many valid datapoints as possible to our dataset. In our example, as there are 2 entities, either one or both can be hidden, yielding 3 datapoints.

Given its inputs and the sender's message, the listener tries to reconstruct the sender's inputs. The agents are jointly trained to minimize a reconstruction loss while minimizing the number of symbols exchanged, as formalized in Section 9.2.2.

9.1.3. Motivations

All the aspects of the task can have a major influence on the learned languages. In this section, we argue that our task is realistic in important aspects.

Our task is to convey semantic annotations of sentences, not words or sentences directly, because using linguistic data as input could be a methodological mistake. Indeed, language-specific typological properties might leak into the artificial languages.[1] We follow this principle, except for our use of θ-roles. They are linguistic abstractions over relation-specific (participant) roles. This limitation is discussed in Section 9.8.2.

In our task, agents have to communicate about a realistic, variable number of entities. We posit that this is a crucial characteristic for argument structure to be natural. Indeed, if humans only ever talked about two entities at once, grammar would be simpler since a transitive construction could be used everywhere. In our dataset, the distribution of the number of entities talked about is directly derived from an English corpus, and, to our knowledge, the distribution of the number of arguments does not vary much across languages. Thus we hope we do not expect a bias towards English typology. In Mordatch and Abbeel [2018]'s and Bogin et al. [2019]'s works, agents also need to predicate. However, the event structure is unrealistic as it is identical across datapoints: the number of arguments is constant and each argument has the same "type" (a landmark, an agent, a color, etc.).

The relation β is observed by both agents. As a consequence, we do not expect artificial sentences to contain the equivalent of a verb. The main reason is that it greatly simplifies the analysis of the artificial languages. Another reason is that in our dataset, relations are coded as categories. Thus, there is no notion of distance between relations which would

[1]For example, if the task was to transmit basic color *terms* instead of, say, color represented as RGB triplets, the choice of a language with only 3 basic color terms vs 11 color terms (as in English) would yield different artificial languages. For one thing, transmitting English color terms would require agents to use more symbols.

enable agents to make interesting generalizations, for instance, that relations similar in some specific aspect share similar argument structures.

We define the *context* as everything that is observed by both agents: the relation and the non-hidden entities. We now justify why agents share context, and why the loss function includes a penalty to minimize the number of symbols sent (cf Section 9.2.2).

First, let us argue that this is realistic. The context is a coarse approximation of the notion of common ground. According to Clark [1996], common ground encompasses the cultural background of the interlocutors, their sensory perceptions, and their conversational history. In theory, the speaker only needs to communicate the information that is not part of the common ground, but transferring more information than needed is not ruled out. However, in practice, humans try to be concise (cf. Grice [1975]'s maxim of quantity). The penalty that we use encourages parsimony. It could be seen as the result from a more general principle governing cooperative social activities [Grice, 1975] or even the whole of human behavior [Zipf, 1949].

To illustrate, consider the following situation. Upon seeing a broken window, one would ask "who/what broke the window?". A knowledgeable interlocutor would answer "John" or "John did". In our setup, the speaker is this knowledgeable person, answering such questions about unobserved entities. The context contains the broken window, and the speaker does not need to refer to it since i) the listener observes it, and since ii) the speaker knows that the listener observes it (via the mask α). While the speaker *could* still refer to the window, the least-effort penalty makes it costly to do so, so the speaker avoids it. Even if the agents do not engage in dialogues but in one-time interactions, the mask α can be interpreted as simulating an inference made by the speaker about the listener's knowledge.

This setup is not only realistic, it is also especially useful for the purpose of analysing the emergent languages. By masking all but one entity, we obtain an artificial *phrase* that denotes a single entity. By masking all but two entities, we obtain an artificial *sentence* relating two entities. The metrics that we introduce rely on our abilities to obtain such phrases and sentences. The concatenability metrics can be seen as measures of systematicity, i.e. how the meaning of phrases is related to meaning of sentences [Szabó, 2020].

Without this setup, one would need to somehow segment sentences into phrases. To our knowledge, the problem has not been addressed in the language emergence literature, but is identified by Baroni [2020]. For instance, applied to English corpora, metrics for quantifying compositionality like Chaabouni et al. [2020]'s disentanglement metrics would tell us that English is not compositional, since single letters are not meaningful.

9.2. Model and objective

We present two Transformer-based [Vaswani et al., 2017] variants of the model of the agents: one that encodes an object-centric bias and one that does not. Before delving into their differences, let us describe their common features.

9.2.1. General architecture

Both the speaker S and the listener L are implemented as Transformers, each of them built out of an encoder Tfm_e and a decoder Tfm_d.

The inputs of the speaker are encoded into a real-valued matrix V^S which differs in the two variants of the model. For now, assume that V^S encodes the speaker's inputs and similarly, that V^L encodes the listener's inputs.

The speaker produces a message m by first encoding its input into

$$H = \text{Tfm}_e^S(V^S), \tag{9.2.1}$$

then auto-regressively decodes the message

$$m_{t+1} \sim q(m_{t+1}|m_{1:t}, I^S, \alpha, \beta) = \text{Tfm}_d^S(M_{1:t}, H)_t$$

with $M_{1:t}$ the sum of positional and value embeddings of the previously decoded symbols $m_{1:t}$.

At train time, the symbol is randomly sampled according to q, whereas at test time, the most likely symbol is picked greedily. If the maximum length n_L is reached, eos is appended to the message and generation stops. Else, the generation process stops when eos is produced. In order to backpropagate through the discrete sampling, we use the Straight-Through Gumbel estimator [Jang et al., 2017, Maddison et al., 2016].

L also embeds the message m into a matrix M', and its decoder produces a matrix O^L:

$$\begin{aligned} H' &= \text{Tfm}_e^L(M'), \\ O^L &= \text{Tfm}_d^L(V^L, H'). \end{aligned} \tag{9.2.2}$$

O^L is then used to predict the presence of the objects as well as all the features of the objects. This computation is slightly different depending on the variant of the models and is detailed below.

Note that Tfm_d^S is invariant with respect to the order of the objects in V^S, since we do not use positional embeddings to create V^S, but rather use the role information directly, as will be explained for each model separately.[2] On the other hand, the message m is embedded

[2]When used without positional embeddings, the encoder of the Transformer is permutation-equivariant, i.e. for any permutation matrix P, $\text{Tfm}_e(PX) = P\text{Tfm}_e(X)$; similarly, the decoder is permutation-invariant in its second argument (the encoded matrix H), i.e. $\text{Tfm}_d(PX) = \text{Tfm}_d(X)$. Permutations are applied to the input matrices, the masks, and the role vectors.

using both token and positional embeddings in M and M', so Tfm_d^S and Tfm_e^L are sensitive to word order.

9.2.2. Loss

The loss function is a sum of two terms, a reconstruction loss and a length penalty.

Reconstruction loss: The input to reconstruct is a *set*, the set of pairs of 18 features and a θ-roles. For each θ-role, we predict the corresponding features as well as whether an object i in this role is present or not, denoted by γ_i.

For a given data point indexed by j, the reconstruction loss is the sum over all objects i

$$l_j = \sum_i -[\log p(I_i^S | I^L, m, \beta) + \log p(\gamma_i | I^L, m, \beta)].$$

Length penalty: As done by Chaabouni et al. [2019], we penalize long messages. This can be seen as an implementation of Zipf [1949]'s least-effort principle. In its simplest form, the penalty is a term $p_j = \lambda |m_j|$ where λ is a hyperparameter, and $|m_j|$ is the number of symbols in the message.

However, we noticed that messages collapse to empty messages early on during training. This is similar to the well-known *posterior collapse*, where the approximate posteriors of latents of sequence-to-sequence VAEs collapse to their priors [Bowman et al., 2016]. We fix the issue by adapting two well-known tricks: Pelsmaeker and Aziz [2019]'s minimum desired rate and Kingma et al. [2016]'s free bits. The penalty term becomes

$$p_j = \mathbf{1}_{l_j < \tau} \mathbf{1}_{|m_j| > n_{min}} (\lambda |m_j|),$$

where $\mathbf{1}$ is the indicator function.

For this term to be non-zero, two conditions need to be fulfilled. Firstly, the reconstruction error must be below τ, which is analogous to a minimum desired rate. This threshold can be set without difficulty to a fraction of the reconstruction error incurred by the listener seeing empty messages. In our case, this average error is 18.6. We randomly choose the threshold in $\{5, +\infty\}$ across runs, where $+\infty$ essentially disables the trick.

Secondly, the penalty is above 0 only if the message contains more than n_{min} symbols. This gives models n_{min} "free" symbols for each datapoint. Without this factor, we found that speakers often utter empty messages (in particular, when a single entity is hidden).

For a given data point indexed by j, the total loss to minimize is the sum $l_j + p_j$. During training, the average is taken over a mini-batch ($n = 128$), while during evaluation, it is taken over the entire test split.

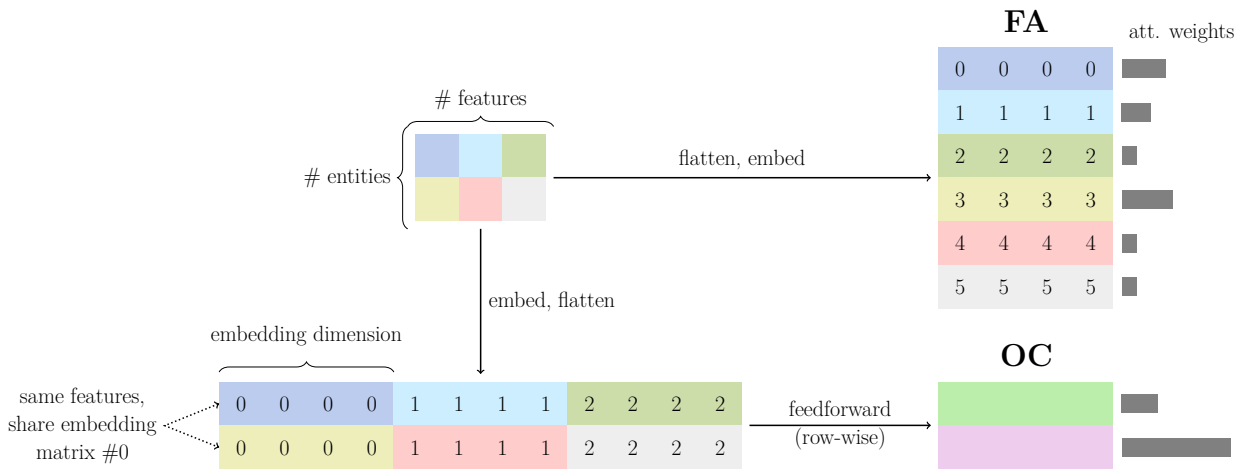

Figure 2. Comparison of flat-attention (FA) and object-centric (OC) variants. The discrete-valued matrices I^L and I^S (upper-left) encode the features of entities. FA turns each datapoint into $(n_{obj} \cdot n_{feat}) \times d$ continuous-valued matrix (with $n_{obj} \cdot n_{feat}$ attention weights), while OC produces a $n_{obj} \times d$ continuous-valued matrix (with n_{obj} attention weights). Numbers index embedding matrices and show weight-sharing. The role information is encoded afterwards and similarly for masking (not shown here).

9.2.3. On the perception of objects

We demonstrate our setup and metrics by comparing a model which is *object-centric* (OC), that is, aware of objects as wholes with properties, to a baseline model (*flat attention*, or FA) which ignores the structure of the inputs.

We follow Gentner [1982], who argued that perception of objects must be a strong, prelinguistic cognitive bias. She gives the example of a bottle floating into a cave. She imagines an imaginary language in which the bottle and the mouth of the cave are construed as a single entity, and argues that this language would be very implausible. Across languages, the two entities seem to always be referred to by separate phrases, hinting at universals in the perception of objects.

More evidence is provided by Xu and Carey [1996]. They showed that infants use spatio-temporal cues to *individuate* objects, i.e. to "establish the boundaries of objects". Only around the start of language acquisition do children start to rely on the properties or kinds of objects to individuate. But could it be exposure to language that drives infant to perceive the properties and kinds of objects? Mendes et al. [2008]'s experiments on apes suggest it is the other way around, i.e. that linguistic input is not necessary to learn to individuate based on property differences. Thus our hypothesis is that the perception of objects as wholes is a prerequisite for natural language to develop.

To implement the OC bias and the FA baseline, we process the inputs in two ways and obtain different V^L and V^S to plug in Equations 9.2.1 and 9.2.2. Embedding the matrices

I^L and I^S gives us real-valued 3-dimensional tensors. But since Transformers consume matrices, we need to reduce the dimensionality of I^S and I^L by one dimension. It is this dimensionality-reduction step that encodes the inductive biases of OC and FA. We tried to minimize the differences between the two models. Figure 2 shows an overview of their differences.

9.2.3.1. Object-centric variant. Let I be either I^S or I^L, where each row I_i represents an object. Each I_i is embedded using a learned embedding matrix Val_j for each feature j, and the result is concatenated, yielding

$$E_i = [\text{Val}_1(I_{i,1})^T; \ldots; \text{Val}_{n_{feat}}(I_{i,n_{feat}})^T].$$

Then, this vector is transformed using a linear function, followed by a ReLU [Nair and Hinton, 2010b] and layer normalisation [Ba et al., 2016]. We obtain $V^{(0)}$, a real-valued $n_{obj} \times d$ matrix with

$$V_i^{(0)} = \text{LN}(\max(WE_i + b, 0)). \tag{9.2.3}$$

As for hidden objects and padding objects, they are represented using a single embedding $V_i^{(0)} = v_h$ directly. A *role embedding* is added to this representation to obtain

$$V_i^{(1)} = V_i^{(0)} + \text{Role}(r_i^S).$$

Finally, V is a $(n_{obj} + 1) \times d$ matrix, where d is the size of embedding vectors. V is $V^{(1)}$ with an additional row vector, the β relation embedding.

The listener cannot distinguish between hidden and padding objects, so the message should encode the roles along with the entities' features.

In order to reconstruct the speaker's inputs, the listener linearly transforms each row vector O_i^L (except the one corresponding to the relation) to produce $p(I_i^S|I^L, m, \beta)$, the joint pmf over the discrete features of object i as well as $p(\gamma_i|I^L, m, \beta)$.

9.2.3.2. Flat attention variant. In FA, the structure of the input – composed of different objects with aligned features – is disregarded. Firstly, the input matrices I^S and I^L, where each row corresponds to a single object, are "flattened". Secondly, there is one attention weight per feature and object *pair*, instead of a single weight per object as in the OC variant. Finally, each embedding matrix is specific to a role and feature *pair*, instead of being specific to a feature.

Formally, let k be the index of a *pair* of object indexed by i and feature indexed by j. Using a k-specific embedding matrix, we obtain

$$V_k^{(0)} = \text{Val}_k(I_{i,j}),$$

with $V^{(0)}$ a real-valued $(n_{obj} \cdot n_{feat}) \times d$ matrix. Again, hidden and padding objects are represented by a special vector $V_k^{(0)} = v_h$. An *index embedding* is added, similar to the role

embedding:

$$V_k^{(1)} = V_k^{(0)} + \text{Idx}(k).$$

As in the OC variant, we obtain V by adding an embedding of the relation β as a row to $V^{(1)}$.

To reconstruct the speaker's inputs, O^L is linearly transformed and to each output vector corresponds a specific feature of a specific object. To predict γ_i, all the output vectors in O^L corresponding to the i-th object are mean- and average-pooled, concatenated and linearly transformed.

9.3. General experimental setup

In the next sections, we review various properties of natural languages, and introduce metrics to quantify these in artificial languages and compare the effect of using OC versus FA on these metrics.

The training set contains 60% of the data, the validation set 10% and the test set the rest. We denote the entire data set by D and denote by D_k the subsets of D composed of examples for which $\sum_i \alpha_i = k$, that is, the examples where k objects are hidden.

All the experiments use the EGG framework [Kharitonov et al., 2021] based on the PyTorch library [Paszke et al., 2019].[3] The neural agents are trained using Adam [Kingma and Ba, 2014].

There is a large number of hyperparameters so we resort to random search [Bergstra and Bengio, 2012].[4] Our hyperparameter search is deliberately broad since we do not know a priori which hyperparameter choices are realistic. We expect to obtain results with high-variance, but a major advantage is that we get more robust conclusions by averaging over unknowns.

We perform linear regressions to predict the value of each metric given a binary variable indicating whether OC is used. When the coefficient for this variable is significantly different from 0 according to a t-test, then OC has a significant effect.[5] Additionally, we consider that the entropy of the messages is a mediator that we control for. For instance, the reconstruction error is indirectly influenced by the vocabulary size and the sampling temperature via the entropy. However, if we observe that OC improves the generalization error, we want to exclude the possibility that this is because OC agents send messages with higher entropy

[3]The proto-role dataset is available here: http://decomp.io/projects/semantic-proto-roles/. The code (including on-the-fly preprocessing of the dataset) is available at https://github.com/tombosc/EGG_f/tree/r1/egg/zoo/vd_reco.

[4]Hyperparameters (uniformly sampled): # Transformer layers $\in$ {1,2,3}, and dimensions $\in \{200, 400\}$, dropout $\in \{0.1, 0.2, 0.3\}$, Gumbel-Softmax temperature $\in \{0.9, 1.0, 1.5\}$, $\lambda \in \{0.1, 0.3, 1, 3, 10\}$, $n_{min} \in \{1, 2\}$, $\tau \in \{5, +\infty\}$. Adam's parameters: $\beta_1 = 0.9$, $\beta_2 \in \{0.9, 0.99, 0.999\}$.

[5]We manipulate data using the pandas package [pandas development team, 2021, McKinney, 2010], and perform linear regression with the statsmodel package [Seabold and Perktold, 2010]. We use HC3 covariance estimation to deal with heteroskedasticity [MacKinnon and White, 1985, Long and Ervin, 2000].

than FA agents, since it should be trivial to also increase the entropy of the FA models by modifying hyperparameters.

We discard models with messages of average length below 1 and above 6. Indeed, when the average length is too small, many messages are empty, and when it is too long, artificial sentences are barely or not longer than artificial phrases. These cases are considered a priori unnatural. This leaves us with 100 out of 136 runs.

Note that the length penalty works as expected. Without the penalty, the messages all contain the maximum number of symbols. With the penalty, the average message length grows as the speaker needs to send more and more information (on D_1: 4.19, D_2: 5.24, D_3: 5.89).

9.4. Generalization performance

Natural languages are often said to be productive and systematic: there is an infinity of utterances which we can understand without having encountered them before (productivity), in particular when we understand constituents of the novel sentence in other contexts (systematicity) [Szabó, 2020]. Do emergent languages exhibit these characteristics? In this section, we study such generalization abilities. We measure how well the listener can reconstruct the inputs when the sender communicates about datapoints unseen at train time.

Firstly, we test our models in distribution. Secondly, we test our models *out of distribution* (OoD), following Lazaridou et al. [2018]. We compute the empirical marginal distributions over the number of hidden entities, the entities, the roles, and the relations. Then, the OoD test set is sampled from these marginals *independently*.

We measure the reconstruction losses on subsets where 1, 2 and 3 entities are hidden for a finer-grained analysis.

Results: Table 1 contains the results. As expect, performance degrades when we evaluate out of distribution. More interestingly, OC models perform better than FA models both in distribution and out of distribution.

However, the performance difference between OC and FA does not tell us much: both OC and FA agents could exchange messages that are structured in very unnatural manners. In the next two sections, we introduce metrics to shed light on how the information about different entities is packaged into a single message.

9.5. Concatenability

In natural languages, the verb encodes the relation while arguments refer to entities, but roles do not have direct equivalents in all languages. They are encoded using three strategies, typically using a mix of strategies within a single language.

	Arch.	1 hidden	2 hidden	3 hidden
iD	FA	6.5 ± 1.6	16 ± 3.6	28 ± 5.4
	OC	6.2 ± 1.9	$14 \pm 3.7^{***}$	$25 \pm 5.6^{**}$
OoD	FA	8.9 ± 2.1	24 ± 3.9	41 ± 5.5
	OC	8.3 ± 2.4	$21 \pm 4.6^{**}$	39 ± 5.9

Table 1. Mean and stdev of test reconstruction loss, in distribution and out of distribution. rows: models; columns: # of hidden entities. OC agents generalize better than FA agents. (*: p-value < 0.05, **: p-value < 0.01)

In analytic languages like English or Chinese, roles are encoded in word order and possibly using adpositions or clitics, but the role does not change the form of the arguments. For example, in sentences (1a) and (1b), the arguments are identical but are placed in a reverse order, so that their roles are inverted too:

(1) a. The old lady walks the dog.

b. The dog walks the old lady.

In more synthetic languages like Russian or Turkish, case markings code for the roles. In Russian, these markings are suffixes on nouns, adjectives, etc., as can be seen in (2a) and (2b):

(2) a. бабушк**а** выгуливает собак**у**.

b. бабушк**у** выгуливает собак**а**.

Finally, in polysynthetic languages (Caucasian languages, Samoan languages, etc.), arguments typically look like those in analytic languages, but the roles are encoded using markers on the verb.[6] Since, in this work, relations are not communicated by agents, there is no artificial equivalent of the verb. Therefore, this strategy cannot emerge and we consider it no further.

Crucially, simple sentences are obtained by concatenating a verb and one or several noun phrases that refer to entities, whether word order matters or word order does not matter and cases are marked.

For a single event, by varying what information is available to the listener through the mask α, we get messages describing two entities in isolation (phrases) as well as messages describing two entities at once (sentences). For example, consider $(I^S, (1,1,0), r^S, \beta)$ drawn from D_2, the subset of the data with two hidden objects. Let g be the function that transforms this speaker's inputs into a message via greedily decoding, and define

$$m^* = g(I^S, (1, 1, 0), r^S, \beta).$$

[6]This presentation is extremely simplified, cf for example Bakker and Siewierska [2009]'s paper for why and how these three strategies generally coexist within a single language.

We obtain the messages sent when L observes the first or the second object in isolation as

$$m^1 = g(I^S, (1,0,0), r^S, \beta),$$
$$m^2 = g(I^S, (0,1,0), r^S, \beta).$$

We define *concatenated messages* to be $m^{12} = m^1 \oplus m^2$ and $m^{21} = m^2 \oplus m^1$, where $\oplus$ is the concatenation operator. This is shown in Figure 1. We define P_2 as the empirical distribution on the subset of D_2 such that neither m^1 or m^2 are empty messages, implying that $m^{12} \neq m^{21}$.

As argued above, in natural languages, m^{12} or m^{21} (or both, if word order is irrelevant) should convey information at least as well as m^*. Denote by $l(m)$ the reconstruction loss incurred by L if L had received the message m, i.e. $l(m) = -\log p(I^S|I^L, m, \beta)$. Then, *concatenability from the listener's point of view* is defined as

$$C^L = \mathbb{E}_{P_2}[l(m^*) - \min(l(m^{12}), l(m^{21}))].$$

When close to 0, on average, one of the two concatenated messages (or both) is as informative as the message actually uttered by the speaker for reconstructing the inputs.

L can correctly reconstruct S's inputs from a concatenated message that S is unlikely to utter. Inversely, a concatenated utterance can be highly likely for S even if L might fail to reconstruct S's input from it. Therefore, there are actually two symmetrical measures of *concatenability*, one from the point of view of S and the other from the point of view of L. A similar proposition was made by Lowe et al. [2019] in the context of interactive games. They have shown the usefulness of distinguishing these two points of view.

The metric is defined similarly on the speaker's side with a slight subtlety. Since sampled messages have a maximum message length of n_L, the probability of a sequence longer than n_L is 0. However, concatenated messages are sometimes longer than n_L. We define q_∞ as the distribution generated by S without the constraint that probable sequences have length below n_L. We denote the conditional log-probability of a message given a certain input by $u(m) = \log q_\infty(m|I^S,\alpha,\beta)$. Then, *concatenability from the speaker's point of view* is defined as

$$C^S = \mathbb{E}_{P_2}[\max(u(m^{12}), u(m^{21})) - u(m^*)].$$

It is close to 0 when, on average, one concatenation of the two messages (or both) has roughly the same probability as the actual message.

To give an intuition, let us go back to our examples. Take the speaker of an hypothetical language, English without verbs. Suppose that this speaker, when exposed to a given input $x^S = (I^S, (1,1,0), r^S, \beta)$, produces a sentence m^* corresponding to (1a), "the old lady the dog". By exposing the speaker to the same input, but by changing the mask to $(1,0,0)$, they produce m^1 = "the lady", while using the mask $(0,1,0)$, they produce m^2 = "a golden

	$C^L \uparrow$	$C^S \uparrow$
FA	-6.1 ± 3.8	-29 ± 13
OC	$-3.2 \pm 2.4^{***}$	-26 ± 15

Table 2. Mean and stdev of concatenability metrics on OC and FA runs. i) OC improves concatenability. Arrows indicate optimal direction. (p-values: *: < 0.05, **: < 0.01, ***: < 0.001)

retriever". C^S compares the log probability of m^* with that of m^{12} = "the lady a golden retriever" and m^{21} = "a golden retriever the lady", whichever is more probable. Since English without verbs is rather concatenable, the speaker judges that m^{12} is roughly as likely as m^* given the inputs. Thus, the value inside the expectation of C^S will be high, close to 0.

Now, take an identical speaker, except that they assign a very high probability to m'^1 = "a shoebox", while the new m'^{12} and m'^{21} are unlikely conditioned on x^S. Then C^S will be low and negative. Perhaps i) "a shoebox" has different semantics when it is used alone in a sentence, as compared to when it is used with a second referent; or perhaps ii) "a shoebox" is never used with another referent in a sentence, and the speaker would use "a lady" instead. In any case, concatenability for this speaker would be low, which corresponds to the intuition that their language is unnatural and unsystematic.[7]

The same illustration holds for C^L, and it can be adapted to show why C^S and C^L should also be high for more synthetic languages.

Results: We measure these metrics on the test set. In our experiments, they always take negative values: the concatenated messages are on average worse than the actual messages. Some models yield values close to 0, but this depends on the choice of hyperparameters.

Table 2 shows that OC largely improves over FA in terms of both C^L and C^S. For instance, the reconstruction losses of OC models go up by 3.1 nats on average when the best concatenated messages are used instead of the actually sent messages. In contrast, FA models incur a loss that is higher by 6.1 nats. Thus, languages obtained using the OC architecture are more natural than those emerging from FA in the sense of concatenability.

9.6. Word order

9.6.1. Importance of word order

Concatenability metrics do not distinguish between the use of word order or some sort of case marking strategy. Since both strategies are found in natural languages, we claim

[7]This example only illustrates the intuition. In reality, it is not straightforward to apply these metrics on natural language, because they require probability distributions for the agents. We could learn models that map back and forth between the semantics and the ground-truth utterances, but the models would add some bias. Moreover, we only have ground-truth utterances for English and any attempts to use machine translation would add some more bias.

	$T^L \uparrow$	$T^S \uparrow$	$RPE \downarrow$
FA	3.4 ± 3.2	10 ± 9	0.48 ± 0.18
OC	$11 \pm 10^*$	15 ± 12	$0.52 \pm 0.12^{***}$

Table 3. Mean and stdev of transitivity metrics and RPE for OC and FA. T^L (T^S) statistics and significance computed on runs scoring C^L (C^S) above median. Arrows indicate optimal direction. OC uses word order more than FA. Controls are discussed in the main text. (p-values: *: < 0.05, **: < 0.01, ***: < 0.001)

that for all natural languages, this metric should be high. But we also want to know what particularly strategy is used, in particular when concatenability is high.

First, note that it is difficult to detect the presence of case markings directly. Even for the simplest forms of morphology, we are hindered by the segmentation problem of identifying the root and the affix, as mentioned in Section 9.1.3.[8]

Yet we can quantify on average how much referential phrases (messages about a single hidden object) encode roles. We train a bigram classifier on the training set and measure its test error, the *Role Prediction Error* (RPE). If there are case markings, this error will be low (but the opposite is not true).

Moreover, we introduce two *transitivity* metrics, to directly measure the importance of word order. T^S is defined as:

$$T^S = \mathbb{E}_{P_2} |u(m^{12}) - u(m^{21})|$$

This metric is 0 if the two concatenated messages are equally probable for S; and it is large if one word order is much more likely than the other for S. Similarly, T^L is defined as

$$T^L = \mathbb{E}_{P_2} |l(m^{12}) - l(m^{21})|$$

and has similar interpretations.

These metrics are only interpretable when concatenability metrics are high enough, so we measured T^S only for runs where C^S is above the median and similarly for T^L.

Results: As can be seen on Figure 3, when transitivity is low and RPE is high, the reconstruction loss is poor (top-left corner), because there is no efficient strategy to encode roles. There is a lot of variance both for OC and FA, but OC models tend to have higher transitivity, both on average and in terms of maximal values. Thus word order is more important for OC runs than for FA runs. This is also confirmed by Table 3.

Table 3 also shows that OC and FA agents have very similar RPE. This means that both encode roles in referential phrases quantitatively similarly. More work is needed to

[8]It is generally even more complicated for several reasons: a lexeme can have several roots, each morpheme can simultaneously encode several semantic properties, and the modification of the root can be non-concatenative [Stump, 2017].

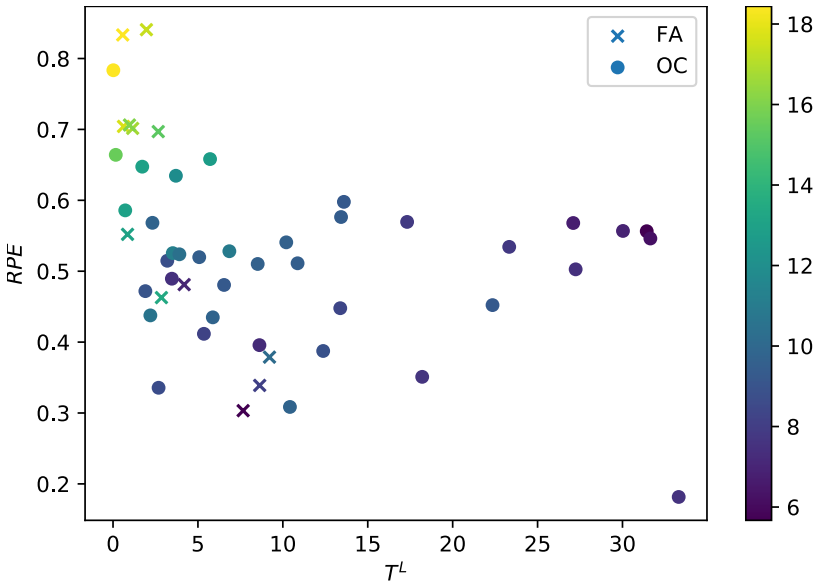

Figure 3. Role prediction error (RPE) as a function of transitivity T^L. Color indicates reconstruction loss. i) (upper-left quadrant) Low T^L and high RPE implies a high reconstruction error, since roles are not encoded properly. ii) OC has higher average transitivity than FA, but similar RPE.

determine how roles are encoded (when they are), that is, if there are traces of morphology or if messages denoting a single entity in different roles are unrelated.

Entities	α	**A**	**B**
$-, 8, 1$	$0, 1, 0$	$24, 79, 25$	105,47
	$0, 0, 1$	$105, 16, 105$	34,34
	$0, 1, 1$	105 , $79, 24$	34 , 47
$-, 8, 5$	$0, 1, 0$	$24, 79, 25$	105,47
	$0, 0, 1$	$19, 24$	18,18
	$0, 1, 1$	47, $79,24,25$	18 ,24
$-, 8, 190$	$0, 1, 0$	$24, 79, 25$	105,47
	$0, 0, 1$	$16, 19$	19
	$0, 1, 1$	16 , 79 ,39, 79	105 ,24
$-, 8, 39$	$0, 1, 0$	$24, 79, 25$	105,47
	$0, 0, 1$	16,44,16,72,2	16,19
	$0, 1, 1$	$44,16$,59, 72	105 , 16

Entities	α	**A**	**B**
$8, 4, -$	1,0,0	79,24,24,79,24	18,1,18
	0,1,0	34,34,15	15,34,15
	1,1,0	34 , 24,79,24,79,24	34,34,34 , 1,18
$8, 61, -$	1,0,0	79,24,79,24,24	18,18,19
	0,1,0	94,54,25,94,72	16,16,25
	1,1,0	94 ,121, 25 , 79,24,79,24	16 , 19 ,24, 19,18
$-, 132, 8$	0,1,0	19,24,19	19,59
	0,0,1	79,24,72	47,71,105
	0,1,1	24, 19 ,123, 19	18,24, 59
$-, 287, 8$	0,1,0	35,19	19,59,16
	0,0,1	79,24,72	47,71,105
	0,1,1	16, 79 , 19,35	24, 19,59,16

Table 4. A sample of messages exchanged about the same entity u_8. Entities: list of entities ("-": no entity; number indicate rank of entity in the dataset; position in the list indicate role: AGENT, PATIENT, MISC). α: mask. **A**, **B**: Messages produced by speakers of models **A** and **B**. Symbols are manually colored to identify phrases (first 2 rows in every block of 3 rows) in artificial sentences (third row in every block). Relations are omitted but are different for each block.

9.6.2. Consistency of word order

To go further, we can study which word orders are favored across different contexts. For every pair of roles such as AGENT and PATIENT, is it the message with the AGENT uttered first that is more likely, or the opposite?

To answer the question, instead of looking at the magnitude of the gap as does T^S, we can count which word orders maximize the gap. By finding the most frequent order, we find for each model the *preference* of the speaker P^S, a binary relation on $\mathcal{R}^2$. For example,

$$\{(\text{AGENT},\text{PATIENT}),(\text{PATIENT},\text{MISC}),(\text{MISC},\text{AGENT})\} \tag{9.6.1}$$

is such a relation. This is very crude, as it does not distinguish the case where AGENT always precedes PATIENT from the case where AGENT precedes PATIENT 51% of the time, but we leave more involved analyses for future work. We define analogously P^L using the reconstruction loss l instead of message probability u.

Results: We compute preferences P^S and P^L for each run. Out of 100 runs, 29 runs have both C^S and C^L higher than their median values, and 23 of these have equal P^S and P^L.

Among all possible relations, some are not transitive, such as equation 9.6.1. However, all the preferences we found are transitive, which is extremely unlikely due to chance. A simple explanation is that transitive relations allows agents to discuss three entities with word order only. However, it does not seem to be universally required by natural languages to have well-defined orders in the presence of many roles. For instance, in English, the use of different prepositions allow for different word order, such as the dative alternation which offers two different orders to talk about three entities.

9.7. Qualitative analysis

One can gain intuition about the metrics by looking at messages exchanged by agents. In particular, we compare two models **A** and **B** which both have relatively high concatenability, but **A** has high transitivity scores whereas those of **B** are low. The chosen models also have relatively close reconstruction loss, so that the messages convey roughly as much information about their inputs.

To simplify, we focus on one entity vector and see how it is transmitted when it is in different roles and in different contexts. Since feature vectors are slightly sparse (with many NA values), vectors which have many NAs are sometimes not conveyed at all (the penalty makes it costly to do so). We search for an entity that appears in many different roles and that is sufficiently not sparse. The 8th most frequent vector (u_8) is the most frequent vector that fits these criteria.

First, let us examine the left-hand side of Table 4, which shows how u_8 is talked about in its most frequent role, the PATIENT role. In both models, u_8 is denoted by the same phrase

very consistently (first rows of each block). Thus the context of u_8 (entities and relation) does not seem to influence the message. This property is sometimes called context-independence [Bogin et al., 2019].

Despite using a large vocabulary of 128 symbols, only a few symbols are used. This is due to the difficulty of discrete optimisation. We were puzzled to find so many common symbols in the two models, but it turns out that the selected models have the same hyperparameters except for the length-penalty coefficient (**A**: $\lambda = 1$, **B**: $\lambda = 10$).

Each last row of each block of three lines shows an artificial sentence, where two entities are hidden. We can see that most symbols in these sentences also frequently appear in phrases that denote individual entities (identified by their colors). Some symbols from phrases are omitted or in a different order in the sentence, but the precise structure of these phrases is out of scope for our work.

A is more consistent in its use of word order than **B**: **A** almost always refers to MISC before PATIENT, whereas the order varies for **B**. This is evidence that the transitivity metrics correctly measure the importance of word order, at least when concatenability is high enough.

On the right-hand side of Table 4, u_8 appears in less frequent roles, and we see much more irregularities. Firstly, the phrases denoting u_8 in isolation are less consistent across different contexts (more context-dependence), even though we find a large overlap of symbols. Secondly, we also found more empty phrases (not shown here). Thirdly, we did not find evidence for a lower transitivity of **B** in these roles, but the sample size was smaller.

9.8. Discussion and limitations

9.8.1. Partial observability and reference

Thanks to our experimental setup and metrics, we avoid the problem of segmentation. However, concatenability and transitivity rely on a crucial aspect of the task, partial observability, which allows us to obtain messages about a single "thing" in isolation. In our case, this "thing" is an entity and role pair, but instead, could it be a single attribute like shape or color, as in simpler referential games used in past research?

Such a setup would be similar to our setup (cf 9.1.2). However, i) there would be no relation β; ii) I^S, I^L and α would be vectors of size n_{feat}; iii) in terms of models, we would use a simple attention mechanism to select a subset of the features to communicate about.

However, we do not think that this setup realistically models real-life communicative situations. Visual properties like shape and color are often perceived simultaneously. If, sometimes, we fail to perceive colors (for example, at night) or shapes (perhaps due to an occlusion), we rarely need to inquire about these attributes. In general, the missing attributes do not particularly matter, but are useful to identify the *kind* of the entity. For example, the

white color and the circular shape of an object tells us that it is a plate, which is useful; but its particular appearance generally does not often matter once it has been categorized. Thus, we generally infer the kind from the observed attributes if possible, or else directly ask for the kind.

By contrast, events are often partially observed, which creates many interrogations. When one observes the consequences of a past action, one often wonders who was the agent that caused it. Similarly, since future events are indeterminate, they are frequently discussed and negotiated. Thus it is frequent to describe events partially.

In sum, the semantics of events are often conveyed partially whereas the semantics of entities are more frequently packaged into the word for a kind. Thus directly transposing this setup to the referential case seems unrealistic. However, perhaps it could be adapted to a discriminative setup [Lazaridou et al., 2017], where the need to convey partial features of objects is clearer.

9.8.2. On θ-roles

As inputs to our models, θ-roles are much more salient than any of the 18 features associated with entities: each θ-role is associated with an entire vector added to the keys and values used by the attention mechanisms (cf. Role and Idx in Sections 9.2.3.1 and 9.2.3.2). Moreover, there are only three of them and they are mutually exclusive. For these reasons, it is easy to attend over each of them, which explains why many artificial agents rely on θ-roles to structure their messages.

These θ-roles are groups of verb-specific roles (sometimes called participant roles). For example, the LOVER, the EATER, and the BUILDER verb-specific roles are clustered into the verb-general AGENT θ-role, while the LOVEE, the EATEE and the BUILDEE roles fall under the PATIENT θ-role. Dowty [1991]'s shows that some θ-roles can be predicted from a small set of features that are mostly related to physical notions of movement and to causality.[9] However, since humans perceive many more features (for example, shapes, colors, textures, etc.), it is not clear why these particular features are preferred to structure the grammars of natural languages.

To answer this question, we might be able to use pretrained unsupervised learning models as feature extractors [Santoro et al., 2017, van Steenkiste et al., 2018, Kipf et al., 2018]. An object-centric model like R-NEM [van Steenkiste et al., 2018] can extract object representations from videos of physically interacting objects. An interaction model like NRI [Kipf et al., 2018] can infer the relations between objects given object representations over time, such that these relations are predictive of how the objects change over time. By combining such models, it may be possible to learn object, relation and role representations

[9]These features are precisely the features that are used in this paper to represent the semantics of the entities, but their meaning is irrelevant in this work.

from videos. We could then use such learned representations as inputs in our communication games to study whether verb-general roles emerge.

9.9. Conclusion

We have presented an experimental setup for studying how probabilistic artificial agents predicate, that is, how they convey that a relation holds between entities. In our daily lives, events are partially observed and predication is used to share information about what is not observed, often in a parsimonious manner. Our task and loss realistically reflect this function of language.

At the same time, this setup allows us to directly study argument structure while ignoring the internal structure of phrases. Indeed, we can easily obtain artificial phrases, i.e. utterances that refer to single entities, as well as artificial sentences, utterances which express the relation holding between different entities. Then, we can study whether and how artificial phrases are systematically composed to form artificial sentences, via our concatenability and transitivity metrics. Thus we completely sidestep the need to segment artificial sentences into phrases, a complicated problem that is unfortunately ignored in previous works.

More precisely, we have argued that all natural languages should have high concatenability, while transitivity is not necessarily high and merely quantifies the importance of word order.

Equipped with this setup and these metrics, we have compared a cognitively plausible architecture that leverages the structure of the inputs into objects with properties (OC) against an implausible baseline that ignores this structure (FA). Object-centric models yield more natural languages in terms of concatenability, while also relying more on word order. Moreover, they generalize better than their implausible counterparts, both in distribution and out of distribution.

These results confirm the importance of the input representations and of the architectures leading to the discretization bottleneck, also reported by Lazaridou et al. [2017] and Guo et al. [2019]. In our experiments, discrete low-dimensional inputs were processed by task-specific architectures. However, we believe that one can use high-dimensional representations obtained from pretrained models, as long as these representations are prelinguistic, as object-centric representations seem to be.

Our methods could be extended to investigate other aspects of sentences. For instance, how would agents convey relations? To answer this question, we could use the representations learned via relational unsupervised learning algorithms as inputs. We could study how different relations are discretized into one or several symbols, perhaps the equivalent of verbs and adverbs. We could also analyze how relation-specific roles cluster in abstract roles (like θ-roles) and structure grammar.

Acknowledgements

Tom Bosc was financially supported for this research by the Canada CIFAR AI Chair Program. We also thank the Mila IDT team for the computational infrastructure, as well as anonymous reviewers and action editors Alexander Clark and Daniel Gildea for their helpful feedback.

Chapter 10

Conclusion

In this thesis, we showed three very different applications of autoencoders.

In the first paper, we experimented with an autoencoder of dictionary definitions that learn word vectors. It can be used to modify existing distributional vectors or to learn to produce embeddings from scratch, only using a dictionary. It has remarkable one-shot learning abilities. Moreover, the learned word vectors are qualitatively different from those learned via distributional methods. They reflect semantic similarity more, while being less organized in terms of association and relatedness.

In the second paper, we showed how to analyze the representations learned by the sequence-to-sequence VAE. The learned representations are generally not capturing global aspects of texts, such as their topic or their sentiment. This limits drastically the usefulness of models for controllable text generation. After correcting some misconceptions about the evaluation of the models, we showed that using simpler architectures with bag-of-word assumptions, or using pretraining, we can learn more global representations.

The third paper deals with language emergence simulations, where an agent, the speaker, has to convey some meaning to another agent, the listener, by transmitting a sequence of symbols. We proposed a new task in which agents need to predicate, i.e. talk about several entities instead of a single one. As a consequence, the speaker needs to form a sentence made of several phrases, instead of a single phrase that refers to a single entity as in reference tasks. We provided metrics to analyze the learned languages that cover two of the three main strategies used by humans to communicate the roles of the entities. We showed that when agents perceive entities as wholes over which they can attend, they communicate more naturally, leveraging word order or indicating the roles directly on phrases.

In the remainder of this chapter, we discuss the relevance of these results to broader problems and questions such as data efficiency, distribution shifts and grounding. We also take a step back to discuss assumptions under the language emergence paper and future work.

10.1. Data efficiency and distribution shifts

In NLP tasks, labelled data is always too few. The set of possible sentences is so large that it is hard to improve supervised learning algorithms without overfitting. Unsupervised and semi-supervised learning methods, such as word embeddings and language models, tackle this problem by leveraging large volumes of unlabelled data. Moreover, statistical NLP methods can be vulnerable to distribution shifts. Language use adapts to a changing world: new words and constructions are introduced, new topics are discussed, new opinions and arguments are voiced. Therefore, NLP models need to be updated and continually learn from new data. To deal with this, meta-learning methods make use of the hierarchical structure of the training data into distinct, related datasets to learn quickly from a few examples. Continual learning algorithms focus on learning continuously while minimizing forgetting.

The dictionary paper blend these different learning paradigms into a single model. The method is supervised, in the sense that each word is associated with a definition, and the penalty matches the word representation to the definition representation. At the same time, it is also unsupervised, since it learns to represent definitions by autoencoding. The encoder is also a one-shot learner, predicting word vectors which are parameters of NLP models. Finally, it is theoretically able to perform continual learning and improve. Indeed, the definition of a new word w can be encoded into a vector added to the embedding matrix of the encoder. The model has improved, since it is now able to better encode definitions that contain w.

In natural language processing as a whole, the distinction between these paradigms is getting blurrier over time. Nowadays, a few pretrained large language models are massively reused by the community. When fine-tuned or prompted, they solve downstream tasks with relatively little labeled data and impressive performance [Devlin et al., 2018, Radford et al., 2019, Brown et al., 2020]. Powerful models such as GPT3 [Brown et al., 2020] can learn to solve new tasks without supervision, simply by being conditioned on the right prompt. In a way, they also continually learn by conditioning as well. They completely subsume and make obsolete the dictionary autoencoder. For instance, they can use a made-up word appropriately, given an invented definition.

10.2. Grounding

While we now know from large language models such as GPT3 that learning purely from text data goes a long way, the issue of grounding remains. *Grounding* an expression usually means associating this expression to stimuli in another modality, such as images, actions in a reinforcement learning setting, etc. [Harnad, 1990]. Cognitive scientists have shown that our experience of the physical world [Lakoff, 1987] and our social interactions [Tomasello, 2010] have tremendous importance for our linguistics skills. Thus, part of the NLP community argues that language understanding requires grounding [Bisk et al., 2020]. There is still

uncertainty regarding this claim. Indeed, the level of understanding required to solve a given NLP task depends on the task, and whether grounding helps or not is an empirical matter for now. Yet, grounding language in other modalities is already incredibly useful, if not to solve purely textual NLP applications, at least to control RL agents [Luketina et al., 2019], to generate images based on text descriptions [Ramesh et al., 2021, Rombach et al., 2022, Ramesh et al., 2022], and many more applications will come. Thus, we can safely assume that it is desirable and research is already firmly oriented towards this goal.

Some words and expressions are particularly abstract and do not seem strongly associated with any sensory perception or motor output. For instance, how are terms like "rights", "jurisprudence" and "settlement" felt and experienced in the body? Using definitions is a possible solution to learn to process these words. This seems to be done implicitly by large language models, and all the more because such abstract words are also frequently defined (within pedagogical documents, for example). It would be interesting to compare the acquisition of such concepts in artificial models using definitions, versus other mechanisms such as analogies and metaphors which might be cognitively more plausible [Lakoff, 1987].

Regardless of how the understanding of such legal terms take place, it must rely on underlying on the understanding of lower-level concepts such as good and bad, rights and duties, rewards and punishments, etc. In turn, it is hard to imagine agents that can understand these simpler terms outside of a social environment [Fillmore, 1976, Lakoff, 1987, Tomasello, 2010]. Similarly, our bodily experience seems to be at the heart of our understanding of language as well [Lakoff, 1987]. Language emergence simulations could be another helpful tool to study how to efficiently ground language in the social and physical world. A strand of language emergence research examines how artificial agents learn to cooperate with other agents, whether artificial or human, using natural language [Lazaridou and Baroni, 2020, Section 4]. The third paper is a study of language emergence with a different objective: to learn more about why natural languages are the way they are. But we hope that the techniques introduced are general enough to also help ground artificial agents and improve their speaking and understanding abilities.

10.3. Language emergence

The method of the third paper can be generalized and summarized as the following recipe: i) find some linguistic phenomenon to study (here, the way the arguments of the verb are positioned around the verb and/or marked by case); ii) create metrics to measure this phenomenon in artificial languages in a way that captures the cross-linguistic diversity of natural languages (concatenability, transitivity); iii) posit the influence of some factor (perception of objects as distinct wholes) on i). Finally, iv) vary the factor iii) and measure how *relatively* more or less natural according to i) and ii) the emergent languages are. This

approach is interesting to test counterfactuals, that is, to simulate experiments that are impossible to realize. As we did in the paper, we can make comparisons between different possible cognitive mechanisms, but we could also test the influence of the input distribution, the objective function, the patterns of interactions between agents, etc.

In the third paper, we implicitly assumed that there are different forms of compositionality. This is also the case in formal semantics to some extent: the functions that encode the semantics of verbs, nouns and adjectives have different types [Winter, 2016]. Similarly, some cognitive linguists argue that the different parts of speech typically stand for different kinds of things in the world, for example, that the prototypical noun phrase correspond to an entity (one of Lakoff [1987]'s central principles). Based on this assumption, we can then posit different causes for these different forms. By contrast, most other works suffer from not clearly defining the linguistic phenomenon of interest in the first place, and talking about compositionality as a general phenomenon. For instance, when agents talk about geometric shapes [Kottur et al., 2017], the expectation is that that they should use referential phrases like "red triangle". To our knowledge, however, the fact that such phrases consist of two different parts of speech (nouns and adjectives) is not discussed. There is no mention of cross-linguistic variations of this two-words pattern either (for example, there could be significant variations in languages with closed class of adjectives [Dixon, 1977]); no metrics that would correspond to these different variations and would be maximized on different natural languages; and no realistic explanation is proposed as to how these classes arise either, and why one attribute is conveyed with a noun (the shape) while the other is conveyed with an adjective (the color). If one follows our recipe, all these points should be addressed.

In a subsequent work-in-progress [Bosc, 2022], we argued that costly communication and partial observability could explain compositionality between adjectives as well. But the two forms of compositionality considered in these two works arise because of different objective functions (reconstruction versus discrimination objectives) and require different perceptual skills (perceive objects as wholes versus perceive objects as having properties or scalar dimensions that are aligned).

One important difficulty remains: costly communication and partial observability are at the same time explanations for the emergence of compositionality *as well as* crucial tools to analyze the messages. In other words, it is difficult to verify that agents communicate less naturally without the communication costs and the partially observed inputs since it is also much harder to interpret their messages when the task does not possess these characteristics.

Another issue is the quality and nature of the inputs that the agents communicate about. In the third article, these inputs consist of sets of objects, and each object is described by a vector. As discussed in Section 9.8.2, this data is produced using annotations of English texts. This might introduce a bias toward English grammar in the emergent languages. One way to tackle this might be to use unsupervised models. Object-centric and relational models

could be used to extract entity and relation representations from videos without supervision; and such representations could be used as inputs. For the study of compositionality between adjectives and nouns, there is a similar need for unbiased data. We think that disentangled representations of entities soon will be obtained without supervision and used as inputs in language emergence simulations. As Zheltonozhskii et al. [2020] show, by clustering unsupervised learning representations of images, without even using linear classifiers, we can recover the labels of images with relatively high accuracy. Such unsupervised class labels could be further exploited by Khemakhem et al. [2020]'s disentanglement method. Their VAE can learn disentangled latent variables when they are conditioned on some (possibly noisy) class label and under some assumptions on the true data distribution. Thus, we might be able to encode videos as sets of segmented entities whose properties are disentangled, and relations between these entities are characterized as well by vectors or categories. Such representations could then be used as inputs to agents in language emergence simulations.

Instead of studying language emergence from scratch, we could also study language evolution. We could design an initial language by hand, and distill it into neural agents. Then, we could optimize various objectives and observe changes in the languages. For instance, we could study *grammaticalization*, that is, how lexical items change and become parts of grammatical rules [Bybee, 2015].

There is also significant overlap between language emergence research in NLP and the Rational Speech Act model [Goodman and Frank, 2016] in cognitive science: both strands of research study language from a functional perspective, highlighting i) the importance of communication costs and ii) the different roles of the speaker and the listener. The respective strengths and limitations of the two approaches should be better understood, perhaps to combine them (cf. for example White et al. [2020]'s work).

Bibliography

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a Broken ELBO. In *International Conference on Machine Learning*, pages 159–168, 2018.

Robert Alfred Amsler. The structure of the merriam-webster pocket dictionary. 1980.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *arXiv Preprint arXiv:1409.0473v7*, 2014.

Dzmitry Bahdanau, Tom Bosc, Stanislaw Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. Learning to compute word embeddings on the fly. *CoRR*, abs/1706.00286, 2017. URL `http://arxiv.org/abs/1706.00286`.

Dik Bakker and Anna Siewierska. Case and alternative strategies: Word order and agreement marking. *Malchukov & Spencer (eds.)*, 2009:290–303, 2009.

Marco Baroni. Rat big, cat eaten! Ideas for a useful deep-agent protolanguage. In *arXiv:2003.11922 [Cs]*, March 2020.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Jean-Louis Binot and Karen Jensen. A semantic expert using an online standard dictionary. In *Proceedings of the 10th international joint conference on Artificial intelligence-Volume 2*, pages 709–714. Morgan Kaufmann Publishers Inc., 1987.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. Experience Grounds Language. In *arXiv:2004.10151 [Cs]*, November 2020. Comment: Empirical Methods in Natural Language Processing (EMNLP), 2020.

Ben Bogin, Mor Geva, and Jonathan Berant. Emergence of Communication in an Interactive World with Consistent Speakers. In *arXiv:1809.00549 [Cs]*, March 2019. Comment: Emergent Communication Workshop @ NeurIPS 2018.

Tom Bosc. Varying meaning complexity to explain and measure compositionality. In *Emergent Communication Workshop at ICLR 2022*, 2022.

Tom Bosc and Pascal Vincent. Auto-Encoding Dictionary Definitions into Consistent Word Embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1522–1532, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1181.

Tom Bosc and Pascal Vincent. Do sequence-to-sequence VAEs learn global features of sentences? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4296–4318, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.350.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1002. URL `https://www.aclweb.org/anthology/K16-1002`.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. Comment: 40+32 pages.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 136–145, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. Multimodal distributional semantics. *Journal of artificial intelligence research*, 49:1–47, 2014.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Joan Bybee. *Language Change*. Cambridge University Press, 2015.

Nicoletta Calzolari. Detecting patterns in a lexical data base. In *Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics*, pages 170–173. Association for Computational Linguistics, 1984.

Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul): 2079–2107, 2010.

Rahma Chaabouni, Eugene Kharitonov, Emmanuel Dupoux, and Marco Baroni. Anti-efficient encoding in emergent communication. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 6293–6303. Curran Associates, Inc., 2019.

Rahma Chaabouni, Eugene Kharitonov, Diane Bouchacourt, Emmanuel Dupoux, and Marco Baroni. Compositionality and Generalization In Emergent Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4427–4442, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.407.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035, 2014.

Martin S Chodorow, Roy J Byrd, and George E Heidorn. Extracting semantic hierarchies from a large on-line dictionary. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, pages 299–304. Association for Computational Linguistics, 1985.

Herbert H Clark. *Using Language*. Cambridge university press, 1996.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167, 2008.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

Thomas M Cover. *Elements of Information Theory*. John Wiley & Sons, 1999.

Mathias Creutz and Krista Lagus. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning* -, volume 6, pages 21–30, Not Known, 2002. Association for Computational Linguistics. doi: 10.3115/1118647.1118650.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Jeffrey De Fauw, Sander Dieleman, and Karen Simonyan. Hierarchical autoregressive image models with auxiliary decoders. *arXiv preprint arXiv:1903.04933*, 2019.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv Preprint arXiv:1810.04805*, 2018.

RMW Dixon. Where have all the adjectives gone? *Studies in Language. International Journal sponsored by the Foundation "Foundations of Language"*, 1(1):19–80, 1977.

David Dowty. Thematic Proto-Roles and Argument Selection. *Language*, 67(3):547, September 1991. ISSN 00978507. doi: 10.2307/415037.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. Retrofitting Word Vectors to Semantic Lexicons. In *arXiv:1411.4166 [Cs]*, November 2014. Comment: Proceedings of NAACL 2015.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. Problems with evaluation of word embeddings using word similarity tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 30–35, 2016.

Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

Jessica Ficler and Yoav Goldberg. Controlling Linguistic Style Aspects in Neural Language Generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, 2017.

Charles J Fillmore. Frame semantics and the nature of language. In *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech*, volume 280, pages 20–32. New York, 1976.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001a.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, pages 406–414, 2001b.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing. 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 240–250, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1021. URL `https://www.aclweb.org/anthology/N19-1021`.

Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, 1994.

Dedre Gentner. Why nouns are learned before verbs: Linguistic relativity versus natural partitioning. *Center for the Study of Reading Technical Report; no. 257*, 1982.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10):2451–2471, 2000.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869*, 2016.

Yoav Goldberg and Omer Levy. Word2vec Explained: Deriving Mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Noah D. Goodman and Michael C. Frank. Pragmatic Language Interpretation as Probabilistic Inference. *Trends in Cognitive Sciences*, 20(11):818–829, November 2016. ISSN 13646613. doi: 10.1016/j.tics.2016.08.005.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

Herbert P Grice. Logic and conversation. In *Speech Acts*, pages 41–58. Brill, 1975.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676, 2019.

Shangmin Guo, Yi Ren, Serhii Havrylov, Stella Frank, Ivan Titov, and Kenny Smith. *The Emergence of Compositional Languages for Numeric Concepts Through Iterated Learning in Neural Agents*. 2019.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1406–1414. ACM, 2012.

Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3): 335–346, 1990.

Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. Lagging Inference Networks and Posterior Collapse in Variational Autoencoders. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rylDfnCqF7`.

Aurélie Herbelot and Marco Baroni. High-risk learning: acquiring new word vectors from tiny data. *arXiv preprint arXiv:1707.06556*, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. Beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a Definition of Disentangled Representations, December 2018.

Felix Hill, Kyunghyun Cho, Sébastien Jean, Coline Devin, and Yoshua Bengio. Embedding word similarity with neural machine translation. *CoRR*, abs/1412.6448, 2014a. URL `http://arxiv.org/abs/1412.6448`.

Felix Hill, Roi Reichart, and Anna Korhonen. SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. In *arXiv:1408.3456 [Cs]*, August 2014b.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. Learning distributed representations of sentences from unlabelled data. In *arXiv Preprint arXiv:1602.03483*, 2016a.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. Learning to Understand Phrases by Embedding the Dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30, December 2016b. ISSN 2307-387X. doi: 10.1162/tacl_a_00080.

Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 2016c.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science (New York, N.Y.)*, 268(5214):1158–1161, 1995.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8): 1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics : a journal of statistics for the physical, chemical, and engineering sciences*, 12(1):55–67, 1970.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

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

James R Hurford. Biological evolution of the Saussurean sign as a component of the language acquisition device. *Lingua*, 77(2):187–222, 1989.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 95–105, 2015.

Daniel Im Jiwoong Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. Denoising criterion for variational auto-encoding framework. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Ozan İrsoy, Adrian Benton, and Karl Stratos. Corrected CBOW performs as well as skip-gram. In *Proceedings of the Second Workshop on Insights from Negative Results in NLP*, pages 1–8, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.insights-1.1.

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *arXiv:1611.01144 [Cs, Stat]*, August 2017.

Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in Psychology*, volume 121, pages 471–495. Elsevier, 1997.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, April 2017. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/E17-2068`.

Daniel Jurafsky and James H Martin. Speech and Language Processing. 2009.

Eugene Kharitonov, Roberto Dessì, Rahma Chaabouni, Diane Bouchacourt, and Marco Baroni. EGG: A toolkit for research on Emergence of lanGuage in Games, 2021.

Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear ICA: A unifying framework. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2207–2217. PMLR, August 2020.

Douwe Kiela, Felix Hill, and Stephen Clark. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2048, 2015.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-Amortized Variational Autoencoders. In *International Conference on Machine Learning*, pages 2683–2692, 2018.

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *arXiv Preprint arXiv:1312.6114*, 2013.

Diederik P. Kingma, Danilo Jimenez Rezende, Shakir Mohamed, and Max Welling. Semi-Supervised Learning with Deep Generative Models. *CoRR*, abs/1406.5298, 2014. URL `http://arxiv.org/abs/1406.5298`.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

Paul R Kingsbury and Martha Palmer. From TreeBank to PropBank. In *LREC*, pages 1989–1993. Citeseer, 2002.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697. PMLR, 2018. Comment: ICML (2018). Code available under https://github.com/ethanfetaya/NRI.

Simon Kirby and James R Hurford. The emergence of linguistic structure: An overview of the iterated learning model. *Simulating the evolution of language*, pages 121–147, 2002.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. *Advances in neural information processing systems*, 28, 2015.

Satwik Kottur, José Moura, Stefan Lee, and Dhruv Batra. Natural Language Does Not Emerge 'Naturally' in Multi-Agent Dialog. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2962–2967, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1321. Comment: 9 pages, 7 figures, 2 tables, accepted at EMNLP 2017 as short paper.

Brenden Lake and Marco Baroni. Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR, July 2018.

George Lakoff. Women, Fire, and Dangerous Things. What Categories Reveal about the Mind, 1987.

Angeliki Lazaridou and Marco Baroni. Emergent multi-agent communication in the deep learning era. In *arXiv Preprint arXiv:2006.02419v1*, 2020.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-Agent Cooperation and the Emergence of (Natural) Language. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. Emergence of Linguistic Communication from Referential Games with Symbolic and Pixel Input. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196. PMLR, 2014.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539.

Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, 2014a.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, 27, 2014b.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015. doi: 10.1162/tacl_a_00134.

Omer Levy, Kenton Lee, Nicholas FitzGerald, and Luke Zettlemoyer. Long Short-Term Memory as a Dynamically Computed Element-wise Weighted Sum. In *arXiv:1805.03716 [Cs, Stat]*, May 2018. Comment: ACL 2018.

Bohan Li, Junxian He, Graham Neubig, Taylor Berg-Kirkpatrick, and Yiming Yang. A Surprisingly Effective Fix for Deep Latent Variable Modeling of Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3601–3612, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1370. URL `https://www.aclweb.org/anthology/D19-1370`.

Jiwei Li and Dan Jurafsky. Do multi-sense embeddings improve natural language understanding? *EMNLP 2015*, 2015.

Tal Linzen. Issues in evaluating semantic spaces using word analogies. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 13–18, 2016.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging Common Assumptions in the Unsupervised

Learning of Disentangled Representations. In *International Conference on Machine Learning*, pages 4114–4124, 2019.

J. Scott Long and Laurie H. Ervin. Using Heteroscedasticity Consistent Standard Errors in the Linear Regression Model. *The American Statistician*, 54(3):217–224, 2000. ISSN 00031305. HC3.

Teng Long, Yanshuai Cao, and Jackie Chi Kit Cheung. Preventing Posterior Collapse in Sequence VAEs with Pooling. *arXiv:1911.03976 [cs, stat]*, November 2019. URL `http://arxiv.org/abs/1911.03976`. arXiv: 1911.03976.

Ryan Lowe, Jakob N. Foerster, Y.-Lan Boureau, Joelle Pineau, and Yann N. Dauphin. On the Pitfalls of Measuring Emergent Communication. In Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor, editors, *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, pages 693–701. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob N Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. In *IJCAI*, 2019.

Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208, June 1996. ISSN 0743-3808, 1532-5970. doi: 10.3758/BF03204766.

James G. MacKinnon and Halbert White. Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *Journal of Econometrics*, 29(3): 305–325, 1985. ISSN 0304-4076. doi: 10.1016/0304-4076(85)90158-7. HC3 paper.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *arXiv Preprint arXiv:1611.00712*, 2016.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

R. Thomas McCoy, Tal Linzen, Ewan Dunbar, and Paul Smolensky. RNNs implicitly implement tensor-product representations. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BJx0sjC5FX`.

Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56–61, 2010. doi: 10.25080/Majora-92bf1922-00a.

Ken McRae, Saman Khalkhali, and Mary Hare. Semantic and associative relations in adolescents and young adults: Examining a tenuous dichotomy. 2012.

Natacha Mendes, Hannes Rakoczy, and Josep Call. Ape metaphysics: Object individuation without language. *Cognition*, 106(2):730–749, 2008.

Yishu Miao, Lei Yu, and Phil Blunsom. Neural Variational Inference for Text Processing. In *International Conference on Machine Learning*, pages 1727–1736, 2016.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013c.

Richard Montague. Universal grammar. *Theoria*, 36(3):373–398, 1970. doi: 10.1111/j.1755-2567.1970.tb00434.x.

Igor Mordatch and Pieter Abbeel. Emergence of Grounded Compositional Language in Multi-Agent Populations. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1495–1502. AAAI Press, 2018.

Henry B Moss, David S Leslie, and Paul Rayson. Using JK fold Cross Validation to Reduce Variance When Tuning NLP Models. *arXiv preprint arXiv:1806.07139*, 2018.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *arXiv preprint arXiv:1706.00374*, 2017.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California, June 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N16-1018`.

Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. 2012.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010a.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 807–814, 2010b.

Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10, 2009.

Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. Definition modeling: Learning to define word embeddings in natural language. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Gary W Oehlert. *A first course in design and analysis of experiments*. 2010.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-Gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1049.

Artidoro Pagnoni, Kevin Liu, and Shangyan Li. Conditional variational autoencoder for neural machine translation. *arXiv preprint arXiv:1812.04405*, 2018.

The pandas development team. Pandas-dev/pandas: Pandas 1.2.3. Zenodo, March 2021.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Tom Pelsmaeker and Wilker Aziz. Effective Estimation of Deep Generative Language Models. *arXiv preprint arXiv:1904.08194*, 2019.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural*

language processing (EMNLP), pages 1532–1543, 2014.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. Comment: NAACL 2018. Originally posted to openreview 27 Oct 2017. v2 updated for NAACL camera ready.

Olivier Picard, Alexandre Blondin-Massé, Stevan Harnad, Odile Marcotte, Guillaume Chicoisne, and Yassine Gargouri. Hierarchies in dictionary definition space. *arXiv preprint arXiv:0911.5703*, 2009.

Victor Prokhorov, Ehsan Shareghi, Yingzhen Li, Mohammad Taher Pilehvar, and Nigel Collier. On the Importance of the Kullback-Leibler Divergence Term in Variational Autoencoders for Text Generation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 118–127, Hong Kong, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5612. URL `https://www.aclweb.org/anthology/D19-5612`.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM, 2011.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents, 2022.

Ali Razavi, Aaron van den Oord, Ben Poole, and Oriol Vinyals. Preventing Posterior Collapse with delta-VAEs. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BJe0Gn0cY7`.

Drew Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. Semantic Proto-Roles. *Transactions of the Association for Computational Linguistics*, 3:475–488, December 2015. ISSN 2307-387X. doi: 10.1162/tacl_a_00152.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *International Conference on Machine Learning*, 2011.

Anna Rogers, Aleksandr Drozd, and Bofang Li. The (too Many) Problems of Analogical Reasoning with Word Vectors. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 135–148, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-1017.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

Aurko Roy and David Grangier. Unsupervised paraphrasing without translation. *arXiv preprint arXiv:1905.12752*, 2019.

Herbert Rubenstein and John B Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. ISSN 1476-4687. doi: 10.1038/323533a0.

Alexander Rush. The annotated transformer. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 52–60, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-2509.

Magnus Sahlgren. The distributional hypothesis. *Italian journal of linguistics*, 20(1):33–54, 2008.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61: 85–117, 2015.

Tobias Schnabel, Igor Labutov, David M Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. 2015.

Raphael Schumann. Unsupervised abstractive sentence summarization using length controlled variational autoencoder. *arXiv preprint arXiv:1809.05233*, 2018.

H Schütze. Dimensions of meaning. In *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing*, pages 787–796, 1992.

Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. Comment: accepted at ACL 2016; new in this version: figure 3.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, May 2014. ISBN 978-1-107-05713-5.

Xing Shi, Kevin Knight, and Deniz Yuret. Why Neural Translations are the Right Length. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2278–2282, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1248. URL `https://www.aclweb.org/anthology/D16-1248`.

Jonas Sjöberg and Lennart Ljung. Overtraining, regularization and searching for a minimum, with application to neural networks. *International Journal of Control*, 62(6):1391–1407, 1995.

Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216, 1990.

Luc Steels. The synthetic modeling of language origins. *Evolution of communication*, 1(1): 1–34, 1997.

Gregory T Stump. Inflection. *The Handbook of Morphology*, pages 11–43, 2017.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

Richard S Sutton and Andrew G Barto. Introduction to reinforcement learning. 1998.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. LSTM Networks Can Perform Dynamic Counting. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54, Florence, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3905. URL `https://www.aclweb.org/anthology/W19-3905`.

Zoltán Gendler Szabó. Compositionality. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2020 edition, 2020.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Julien Tissier, Christophe Gravier, and Amaury Habrard. Dict2vec: Learning Word Embeddings using Lexical Dictionaries. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 254–263, 2017.

Michael Tomasello. *Origins of Human Communication*. MIT press, 2010.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, 2010.

Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural Discrete Representation Learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6306–6315. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7210-neural-discrete-representation-learning.pdf`.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. Blocks and fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619, 2015. URL `http://arxiv.org/abs/1506.00619`.

Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions. In *International Conference on Learning Representations*, 2018. Comment: Accepted to ICLR 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103. ACM, 2008.

Ivan Vulić and Nikola Mrkšić. Specialising word vectors for lexical entailment. *arXiv preprint arXiv:1710.06371*, 2017.

Lilian Weng. Flow-based deep generative models. *lilianweng.github.io*, 2018.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Julia White, Jesse Mu, and Noah Goodman. Learning to Refer Informatively by Amortizing Pragmatic Reasoning. In *Proceedings of the 42nd Annual Meeting of the Cognitive Science Society*, 2020. Comment: Accepted to CogSci 2020.

Anna Wierzbicka. *Semantics: Primes and universals: Primes and universals*. Oxford University Press, UK, 1996.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Charagram: Embedding words and sentences via character n-grams. *arXiv preprint arXiv:1607.02789*, 2016.

Yoad Winter. *Elements of Formal Semantics: An Introduction to the Mathematical Theory of Meaning in Natural Language*. Edinburgh University Press, 2016.

Fei Xu and Susan Carey. Infants' metaphysics: The case of numerical identity. *Cognitive psychology*, 30(2):111–153, 1996.

Peng Xu, Yanshuai Cao, and Jackie Chi Kit Cheung. Unsupervised controllable text generation with global variation discovery and disentanglement. *arXiv preprint arXiv:1905.11975*, 2019.

Yadollah Yaghoobzadeh and Hinrich Schütze. Intrinsic subspace evaluation of word embedding representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 236–246, Berlin, Germany, August 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P16-1023`.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3881–3890. JMLR. org, 2017.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf`.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1061. URL `https://www.aclweb.org/anthology/P17-1061`.

Evgenii Zheltonozhskii, Chaim Baskin, Alex M. Bronstein, and Avi Mendelson. Self-supervised learning for large-scale unsupervised image clustering. *CoRR*, abs/2008.10312, 2020.

Zachary Ziegler and Alexander Rush. Latent Normalizing Flows for Discrete Sequences. In *International Conference on Machine Learning*, pages 7673–7682, 2019.

George Kingsley Zipf. Human behavior and the principle of least effort. 1949.

Chapter A

Appendix of the first article

A.1. Lists of electronic dictionaries

Electronic monolingual dictionaries are rather widespread compared to machine-readable lexical databases. Wiktionary[1] is a collaborative online dictionary, where 79 languages have more than 10,000 entries and 42 languages have more than 100,000 entries. BabelNet [Navigli and Ponzetto, 2012] aligns Wikipedia and WordNet to automatically obtain definitions ("glosses" in BabelNet terminology).

Alternatively, Wikipedia provides a list of monolingual dictionaries.[2] One can also translate "dictionary" into the desired language and look up this term. For example, there are dictionaries for Friulian and Frisian, which are minority languages spoken by less than a million speakers.[3]

	Development		Similarity			Relatedness				
	SV-d	MENd	SL999	SL333	SV-t	RG	SCWS	MENt	MT	353
word2vec	28.2	73.5	36.5	14.2	23.0	76.4	64.2	74.6	64.0	**68.5**
retrofitting	32.8	**74.4**	40.2	20.7	28.2	**84.0**	**65.3**	**77.7**	**65.6**	62.9
dict2vec	36.5	67.9	41.0	24.2	32.0	74.1	61.7	68.1	57.9	64.8
Hill	29.4	62.4	31.9	17.2	20.0	67.1	53.6	62.9	52.6	50.7
AE	33.0	45.6	34.7	24.1	30.3	71.5	49.3	45.5	38.4	42.5
CPAE-P ($\lambda = 16$)	**39.3**	63.1	**45.2**	**31.5**	**37.4**	69.6	59.3	60.7	50.2	58.5

Table 1. Improving pretrained embeddings computed on a large corpus. Spearman's correlation coefficient $\rho \times 100$ on benchmarks. Same as Table 2 but word2vec is trained on the entire Wikipedia dump. Dict2vec fails to improve. Retrofitting especially improves relatedness, while CPAE improves similarity.

[1] https://www.wiktionary.org/

[2] https://en.wikipedia.org/wiki/List_of_dictionaries_by_number_of_words

[3] https://taalweb.frl/wurdboekportaal
http://www.arlef.it/grant-dizionari-talian-furlan/htdocs/gdbtf.pl

A.2. Data

The dump of Wikipedia that we have mentioned in Section 5.6 is the dump from the 14th June 2014. Although this dump is not available anymore online, results should be replicable with newer dumps.

A.2.1. Split dictionary setting

We briefly describe the algorithm used to split the dictionary. If we had used a regular dictionary instead of WordNet, we could have randomly distributed words into a train, validation and test split. However, by doing so, we would have put synonyms in two different splits, which could be a weak form of a test-set leak. On the other hand, WordNet has sets of synonyms, called synsets, where two words in a synset necessarily share at least one definition.

We create batches of words which definitions do not overlap across batches with the following algorithm: First, we create maps from words to their definitions and their converse. We will create batches indexed by i, and we now describe how to build the batch i. We create a set of definitions that is initialized as a singleton containing a random definition, $C_i = \{d\}$. We instantiate another set $S_i = \emptyset$ that will contain all the words which definitions overlap with at least one other word of the set. We iterate over the set C_i and pop a definition. Then, we go through all the words that possess that definition and add them to S_i, while we also add all the other definitions of these words to C_i. We keep iterating over C_i until it is empty. Then we start again the process on a new batch with a new $C_{i+1} = \{d'\}$ where d' has never been added to a C_i before.

Then, we order the batches S_i by the number of words they contain. We choose the largest batch to be the training set. It can be seen as a large subset of the vocabulary that is "central", in a way. By construction, it contains highly polysemous words and frequent words, as shown in Table 2.

Although our construction method is very different, it is slightly similar in spirit to the grounding kernel presented by [Picard et al., 2009]. They report that words in the grounding kernel are more frequent, as we do for the training set. It is also related to the concept of semantic primes, a subset of the lexicon from which all other words can be defined [Wierzbicka, 1996].

A.3. Hyperparameter search

All embeddings are of size 300 for the small Wikipedia dump experiments and 400 for the large Wikipedia dump experiments.

	# defs	# defs / # words	Avg. counts
Train	36,722	2.87	11.40
Valid	2,109	1.81	4.07
Test	128,223	1.13	1.92

Table 2. Statistics of the split dictionary. By construction, the train set contains much more frequent and polysemous words than the train set. The average counts are geometric averages of the smoothed counts of words, computed on the first 50M tokens of the Wikipedia dump.

A.3.1. GloVe

GloVe is run only on the definition corpus because we found word2vec to be superior. We have fixed the number of epochs to be 50. The window size varies between {5,7,9,11,15,**20**} and x_{max} in {0,5,**20**,100}, where the selected hyperparameters are in bold.

A.3.2. Word2vec

We have used gensim implementation.[4]

On definitions, we do a hyperparameter search on the window size in $\{5, \mathbf{15}\}$ and on the downsampling threshold in $\{0, \mathbf{0.1}, 0.01, 0.001, 0.0001\}$, and we also choose from the skip-gram or CBOW variant, where the skip-gram variant is selected.

On the small training corpus, we only use the skip-gram model and choose the number of iterations in $\{5, \mathbf{30}\}$, the window size in {3,**5**,7,10}, and the downsampling threshold in $\{0, 0.1, 0.01, \mathbf{0.001}, 0.0001\}$, where the selected hyperparameters are boldened.

On the full training corpus, we have used the same hyperparameters except that the number of iteration is reduced to 5, and we have filtered out words that appear less than 50 times.

A.3.3. AE, CPAE, Hill's model

We train the AE and CPAE models with Adam [Kingma and Ba, 2014] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a learning rate of $3 \cdot 10^{-4}$, a batch size of 32. In the settings where we use part of the dictionary, we use early stopping: every 2,000 batches, we compute the mean cost on the validation set and stop after 20000 batches without improvement. On the full dictionary, where we do not have a validation set, we train for 50 epochs.

CPAE and AE models always use a reconstruction cost, so $\alpha = 1$. The λ parameter that weights the cost of the consistency penalty varies in {1,2,4,8,16,32,64}, and the values chosen during model selection on the development sets is indicated in the tables.

[4] `https://radimrehurek.com/gensim/`

A.3.4. Retrofitting

We have used the original implementation.[5] We have tuned the hyperparameter α that controls the proximity of the retrofitted vector to the original vector by grid-search: $\alpha \in \{0.25, 0.5, 1, 2, 4\}$. The selected value is $\alpha = 0.5$ for both the small and the large corpora.

A.3.5. Dict2vec

We have used the original implementation.[6] Dict2vec has many hyperparameters. We fixed $K = 5$, where the K closest neighbours to each word in the original embeddings are promoted to form a strong pair, as well as $n_s = 4$ and $n_w = 5$, the number of pairs to sample. We run a grid-search over the hyperparameters, the coefficients that weight the strong and weak pairs importance in the auxiliary cost: $\beta_s \in \{0.4, 0.6, 0.8, 1.0, 1.2, 1.4\}$ and $\beta_w \in \{0.0, 0.2, 0.4, 0.6\}$. In the smaller data regime (small dump), $\beta_s = 1.2$ and $\beta_w = 0$, and in the larger data regime (the entire Wikipedia dump), the model selection procedure picks $\beta_s = 0.8$ and $\beta_w = 0.2$. When focusing on the similarity relation, it seems better not to use weak pairs, or at least to weight them very low.

A.4. Improving pretrained embeddings on the full Wikipedia dump

The scores of word2vec are not really improved by using the much larger full Wikipedia dump, so we increase the size of the embeddings from 300 to 400. The results are given in Table 1.

The trends are similar to what we have observed on the first 50 million tokens of Wikipedia. However, dict2vec seems a bit better and improves over retrofitting in similarity.

[5] `https://github.com/mfaruqui/retrofitting`
[6] `https://github.com/tca19/dict2vec`

Chapter B

Appendix of the second article

B.1. On the use of KL annealing, the choice of the free bits flavor and resetting the decoder

Li et al. [2019] evaluated their models in the SSL setting (Section 3.3 of their paper). However, their experimental setting is not very rigorous. In the case of the 100 labeled examples, hyperparameter selection is done on a very large validation set of 10000 examples. However, the validation set here should be seen as nothing more than a split of the training data dedicated to optimising hyperparameters. In the words of Cawley and Talbot [2010], "model selection should be viewed as an integral part of the model fitting procedure". Besides methodological issues, we run our own hyperparameter search on the Yelp dataset to properly disentangle the effects of KL annealing, the free bits method and verify the importance of resetting the decoder. We use the semi-supervised learning setting presented in Section 7.5 to evaluate the learned encoders.

B.1.1. The free bits technique and variants

The *original* free bits objective [Kingma et al., 2016] is the following modification to the KL term:

$$\sum_{j}^{K} \max(\frac{\lambda}{K}, KL(q_j(z_j|x)||p_j(z_j)))$$

where indices denote components. In this formulation, each component of the multivariate normal is allowed to deviate from the prior by a small amount. Instead, in the δ-VAE formulation, one component can use of all the λ free bits and the rest of the components can collapse to the prior. This is the variant called δ, used throughout the paper:

$$\max(\lambda, KL(q(z|x)||p(z))).$$

FB	λ	Ann.	F1(5)	F1(50)	F1(500)	F1(5000)	F1(all)	KL
O	2	10	$53.3\pm^{5.5}_{3.3}$	$69.8\pm^{1.8}_{1.3}$	$73.6\pm^{0.2}_{1.7}$	$74.0\pm^{0.1}_{1.8}$	$73.6\pm_{1.1}$	$5.27\pm_{0.47}$
O	2	0	$51.8\pm^{4.8}_{6.7}$	$62.7\pm^{2.5}_{3.8}$	$67.0\pm^{0.4}_{5.6}$	$67.5\pm^{0.1}_{5.8}$	$66.9\pm_{2.7}$	$2.58\pm_{0.46}$
δ	2	10	$51.7\pm^{4.6}_{4.7}$	$64.5\pm^{1.9}_{6.7}$	$68.3\pm^{0.4}_{7.3}$	$69.1\pm^{0.2}_{6.7}$	$68.4\pm_{3.3}$	$2.5\pm_{0.24}$
δ	2	0	$58.7\pm^{5.5}_{3.2}$	$74.0\pm^{2.7}_{4.4}$	$78.1\pm^{0.3}_{4.1}$	$78.6\pm^{0.1}_{4.3}$	$78.6\pm_{1.9}$	$2.27\pm_{0.02}$
O	8	10	$60.0\pm^{6.0}_{8.7}$	$77.5\pm^{1.2}_{2.2}$	$80.8\pm^{0.3}_{4.1}$	$81.2\pm^{0.1}_{4.2}$	$81.2\pm_{2.1}$	$10.67\pm_{0.44}$
O	8	0	$60.2\pm^{7.3}_{4.7}$	$77.7\pm^{2.0}_{2.6}$	$81.4\pm^{0.3}_{2.2}$	$81.7\pm^{0.1}_{2.2}$	$81.5\pm_{0.9}$	$9.48\pm_{0.08}$
δ	8	10	$57.6\pm^{7.6}_{4.2}$	$76.3\pm^{1.4}_{1.1}$	$80.3\pm^{0.3}_{3.0}$	$80.8\pm^{0.1}_{2.9}$	$80.3\pm_{1.0}$	$8.21\pm_{0.07}$
δ	8	0	$60.4\pm^{4.1}_{3.6}$	$80.0\pm^{1.3}_{3.0}$	$82.7\pm^{1.0}_{0.9}$	$83.3\pm^{0.1}_{2.3}$	$83.5\pm_{0.8}$	$8.12\pm_{0.02}$

Table 1. δ-VAE-style free bits with no KL annealing delivers the best SSL performance and the KL value closest to the desired rate. *Ann.*: 0: no annealing, 10: anneal for 10 epochs; *FB*: free bits type; *F1(n)*: F1-score in the n data-regime; *KL*: rate obtained after training.

Other modifications of the free bits technique include the use of a variable coefficient in front of the KL term [Chen et al., 2016], the target rate objective in Alemi et al. [2018], minimum desired rate [Pelsmaeker and Aziz, 2019], etc. A comparison of all these methods is out of the scope of this paper and the δ variant satisfies our only requirement: the rate should be close to the desired rate.

B.1.2. KL annealing and the original free bits method higher the rate

Our hypotheses are:

- KL annealing aims at fixing the posterior collapse and is therefore redundant with the free bits,
- KL annealing performs this role by increasing capacity inconsistently across models, making them harder to compare,
- the original free bits formulation impose the unnecessary constraint that the free bits should be balanced over all components.

To study the influence of the free bits variant as well as of KL annealing, we use the same experimental protocol as described in Section 7.5. To save computations, we fix $d = 16$. We do not perform model selection on the desired rate λ in order to see which methods yield the rates that are closest to the desired rate. Table 1 shows these hypotheses are correct. Therefore, all the experiments in the paper use the δ variant without annealing.

In Li et al. [2019]'s work, the original, per-component variant of the free bits might have been chosen because it trivially maximizes a metric called *active units* (AU). However, to our knowledge, there is no evidence that this metric should be maximized, neither theoretical nor empirical.

Reset.	λ	F1(5)	F1(50)	F1(500)	F1(5000)	F1(all)	KL
N	2	$51.0\pm^{4.2}_{5.6}$	$61.3\pm^{2.0}_{9.2}$	$65.6\pm^{0.5}_{9.2}$	$66.2\pm^{0.1}_{9.5}$	$65.2\pm_{4.9}$	$2.36\pm_{0.15}$
Y	2	$58.7\pm^{5.5}_{3.2}$	$74.0\pm^{2.7}_{4.4}$	$78.1\pm^{0.3}_{4.1}$	$78.6\pm^{0.1}_{4.3}$	$78.6\pm_{1.9}$	$2.27\pm_{0.02}$
N	8	$57.4\pm^{5.6}_{2.4}$	$73.4\pm^{1.5}_{7.3}$	$77.2\pm^{0.3}_{6.6}$	$77.5\pm^{0.1}_{6.7}$	$77.4\pm_{2.6}$	$8.23\pm_{0.08}$
Y	8	$60.4\pm^{4.1}_{3.6}$	$80.0\pm^{1.3}_{3.0}$	$82.7\pm^{1.0}_{0.9}$	$83.3\pm^{0.1}_{2.3}$	$83.5\pm_{0.8}$	$8.12\pm_{0.02}$

Table 2. Resetting the decoder brings very noticeable gains on all data-regimes and with different rates. Yelp dataset, δ-VAE free bits, no KL annealing. For columns interpretations, see Table 1.

Dataset	Splits size	Label	$\lvert\mathcal{Y}\rvert$	$H[Y]$	NLL
AGNews	110/10/10	Topic	4	1.39	128.77±0.21
Amazon	100/10/10	Sent.	5	1.61	82.90±0.10
Yahoo	100/10/10	Topic	10	2.30	81.91±0.36
Yelp	100/10/10	Sent.	2	0.67	34.60±0.28

Table 3. Datasets characteristics. $|\mathcal{Y}|$: number of different labels. $H[Y]$: entropy of labels. NLL: mean negative log-likelihood of LSTM baseline models (std. over 3 runs). Splits size: train/valid/test sizes in thousands.

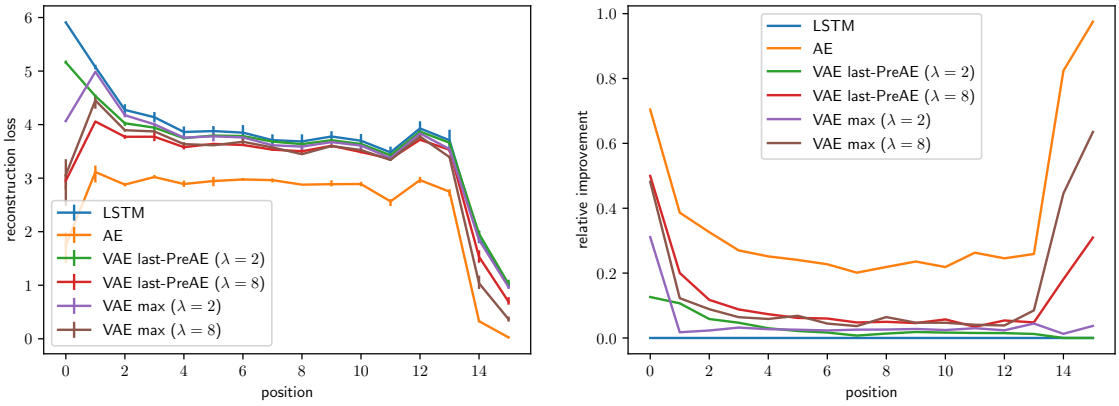

Figure 1. Reconstruction loss as a function of word position on the AGNews dataset. See Figure 1.

B.1.3. On the importance of resetting the decoder after pretraining

Li et al. [2019] proposed to pretrain an *AE* with a reconstruction loss only. Then, the parameters of the decoder are re-initialised and the (modified) KL term is added to the objective. Since it is not very clear why it would be useful, we studied the impact of this choice. Table 2 shows that it is is crucial.

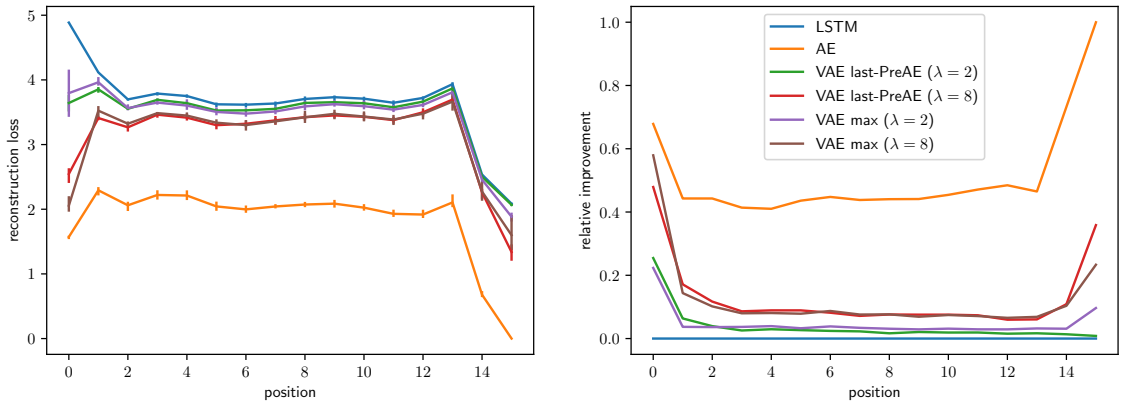

Figure 2. Reconstruction loss as a function of word position on the Amazon dataset. See Figure 1.

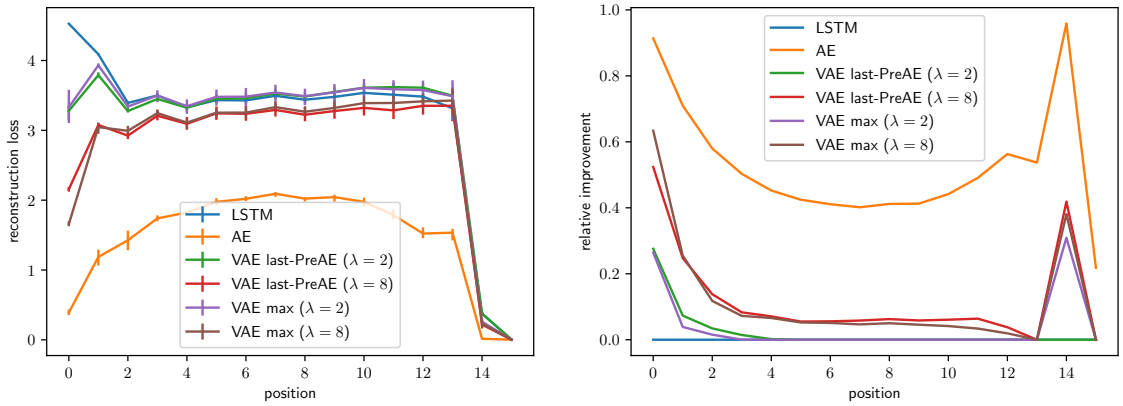

Figure 3. Reconstruction loss as a function of word position on the Yelp dataset. See Figure 1.

B.2. Further evidence for memorization

B.2.1. Plots on other datasets

Figures 1, 2, and 3 show the reconstruction loss and the relative improvement on other datasets.

On the Yelp dataset, the penultimate token is a punctuation mark which is always followed by the end-of-sentence token, so predicting its position is equivalent to predicting the sentence length. That is why the peak at the end occurs before the last token. Moreover, on Yelp, the situation is worse with $\lambda = 2$: between positions 6 and 13, not only is there no improvement, but the reconstruction is higher than that of the baseline.

B.2.2. Tracing back reconstruction gains to words

If words in a document were independently modeled, any improvement in reconstruction at a certain position would indicate that information about the word in that position were encoded in the latent variable. However, words are far from being independently predicted, so how can we trace back the information to the encoder?

First, any latent information related to the first word should not yield any improvements on the prediction of the second word, because the decoder is recurrent and trained using teacher forcing, i.e., conditioned on the true first word, so that information would be redundant. However, information related to the second word in the latent variable can help the decoder predict the first word. Therefore, gains in position i can only be attributed to information pertaining to the words in positions $\geq i$.

Second, the correlation between words in two positions decreases as the distance between these words grow. In effect, information pertaining to the second word yields more gains on the second word than on the first word. From these two facts, we conclude that gains for a position i mostly comes from information about the word in position i itself.

It is harder to draw conclusions about the reconstruction gains at the end of the sentences. Encoding the identity of the last token is only useful if the position of this token (i.e., sentence length) is also encoded. So if the stronger claim that the identity of the last tokens is encoded is correct, sentence length is probably encoded as well. Moreover, this last token is often a punctuation sign that is generally easy to predict given the sentence length. So even if it is memorized, it probably does not matter much with regards to generation as discussed in Section 7.3.2.

B.2.3. Reconstruction and memorization

To study the concrete impact of this observation for generation, we encode and decode test documents using the *last-PreAE* variant.[1] Then, we compute the ratio of documents for which the first word in the sources and in the reconstructions match and similarly, how often the sources and their reconstructions have the same number of words.

We compare these with scores obtained by a baseline model that outputs the most frequent first word given the label and the most common document length given the label. This baseline mimicks the behavior of a hypothetical VAE which would encode the labels of the documents (topic or sentiment) perfectly and nothing more.

Results in Table 4 show that with the *last-PreAE* the first words are reconstructed with much higher accuracy than if the latent vector only encoded the label. On the last two datasets, it recovers the first words on more than half of the documents whereas the baseline only recovers the first words between 11.3 and 14.1% of the time. Accurate encoding of the

[1] $\lambda = 8$, $d = 16$, beam search with beam size of 5.

Dataset	*last-PreAE* 1st (%)	*last-PreAE* Len. (%)	Clf. given label 1st (%)	Clf. given label Len. (%)
AGNews	29.6±1.1	3.6±0.1	12.9	4.8
Amazon	42.4±2.3	13.0±1.6	14.0	0
Yahoo	56.6±1.0	17.1±1.1	11.3	4.9
Yelp	53.0±0.5	33.7±1.7	14.1	9.7

Table 4. The latent variables encode more information than the label alone, in particular, information that allows to retrieve the first word and the document length with high accuracy.

number of words seems less systematic than the encoding of the first few words. For example, on AGNews, the sentence length is recovered less often than our baselines. The encoding of the sentence length is more pronounced on datasets with small documents like Yahoo and Yelp.

B.3. Training procedure

B.3.1. Grid search

The target rates λ are chosen to be higher than the entropy of the labels of the documents (Table 3) as we assume that the latent variable should at least capture the annotated label. Indeed, $\lambda = 2$ nats is enough to store the labels of all datasets without any loss, except Yahoo which has an entropy of 2.3 whereas $\lambda = 8$ nats suffices to capture much more information than needed to store the labels on all datasets. Moreover, these rates are chosen to be much smaller than the reconstruction loss of the baselines because of the technical difficulty of increasing the rate without degrading the log-likelihood explained above.

The latent vector dimension d is either 4 or 16. Recall that our representations are evaluated on downstream tasks with very limited data in some cases (as little as 5 examples per class), so we need a small enough dimension of latent vector to be able to learn. We suppose that $d = 4$ will be favored for the 5 or 50 examples per class regime while $d = 16$ could be more efficient above this, but we leave this choice to the model selection procedure.

B.3.2. Constant hyperparameters

All the runs are trained using SGD with a learning rate of 0.5 and gradients are clipped when their norms are higher than 5. We use the following early stopping scheme: at every epoch, if there has not been improvements on the validation error for two epochs in a row, the learning rate is halved. Once it has been halved four times, the training stops.

All the LSTMs have hidden state size of 512 and use a batch size of 64. No dropout is applied to the encoders. The LSTM decoders use dropout ($p = 0.5$) both on embeddings and

on the hidden states (before the linear transformation that gives logits). Similarly, dropout is applied to the representation before the linear transformation that gives the logits for the Unigram decoder. Word embeddings are initialized randomly and learned.

B.3.3. Computing infrastructure and average runtime

We performed training and evaluations of the models on a cluster containing a hundred of GPUs with various specifications (NVIDIA Tesla k80, Titan X, Titan Xp, etc.). Given that all the datasets have roughly 100000 training examples (cf. Table 3) and that neural networks are trained with BPTT [Werbos, 1990], the training time mostly depends on the average sentence length and the vocabulary size. Pretraining schemes (*PreAE*, *PreUni* and *PreLM*) require the training of two models. Roughly, the training time of a single model (pretraining or final) varied between 1 hour and 6 hours.

To be more specific, the best baselines and our best variants use pretraining phases (*PreAE* from Li et al. [2019] and *PreUni*, respectively). *PreUni* is faster because the first training phase uses a non-recurrent decoder, and the second training phase does not backpropagate and does not update the encoder. However, *PreAE* does not require pretraining for each desired target rate λ, unlike our approach. Overall, the approaches have comparable runtimes.

Some of our models offer an interesting compromise: *BoW-max-LSTM* with no pretraining and a simpler architecture is probably the fastest, yet outperform the *PreAE* baselines.

B.4. Related work

B.4.1. Related models

The models that we use are similar to already proposed models.

The NVDM model of Miao et al. [2016] is precisely *BoW-max-Uni*.

Zhao et al. [2017] proposed to use two reconstruction losses: the regular reconstruction loss given by the recurrent decoder and an auxiliary loss computed from a unigram decoder. In comparison, our *Uni* models are trained in two steps: the encoder is trained jointly with the unigram decoder, then the decoder is thrown away and we train a recurrent decoder using the fixed encoder. This way, one decoder cannot dominate the other and we do not have to deal with an additional hyperparameter to weight the two losses.

Instead of using an auxiliary *loss*, we have an auxiliary *decoder* that is only used for the purpose of training the encoder. This method was presented by De Fauw et al. [2019] for training generative models of image. There is a slight difference: they use a feedforward auxiliary decoder to produce different probability distributions for all the pixels, whereas our unigram probability distribution is the same for all words of a document. This modification allows us to deal with varying lengths of documents.

Finally, the *PreLM* training procedure is related to large LM pretraining in the spirit of contextualized embeddings [Peters et al., 2018] and its successors. Note, however, two differences. Firstly, we do not use external data and stick to each individual training set, because the goal is not to evaluate transfer learning abilities. Secondly, we do not fine-tune the entire encoder, but only learn the linear transformations L_1 and L_2 that produce the variational parameters, to make sure that the VAE objective will have no impact on the extraction of features.

B.4.2. Methods and evaluations

In their analysis of the semi-amortized VAE, Kim et al. [2018] use several saliency measures (defined as expectations of gradients) to determine which words influence the latent variable, or are influenced by it. Using these measures, they noticed that the beginning of the sentence and the end-of-sentence token have a large influence on the variable. Our method is very similar, but slightly simpler and directly interpretable in terms of quantity of information (in nats).

Ficler and Goldberg [2017] learn *LSTM-LMs* conditioned on labels that describe high-level properties of texts. Among others, they want to verify that generated texts exhibit the same properties as the conditioning labels. For instance, when the *LSTM-LM* is conditioned on positive sentiment value, the generated texts should also exhibit a positive sentiment. To check that the conditioning variables and the generated texts are consistent, they use the following procedure. First, they extract information about the various documents using heuristics or with the help of annotators. Then, they learn *LSTM-LM*s conditioned on these labels. Finally, they quantify the ratio of generated samples which have the same labels than the conditioning labels, either by applying the same heuristics again to the generated samples or by asking human annotators once more. Our evaluation in Section 7.6 is similar; we simply replace the heuristics and the human annotators with classifiers learned on ground-truth data.

McCoy et al. [2019] trained autoencoders with different combinations of encoders and decoders (unidirectional, bidirectional or tree-structured) and decomposed the representations learned by the encoders using tensor product representations [Smolensky, 1990]. They find that decoders "largely dictate" the way information is encoded. This is in line with our own conclusions. An important difference between our works is that they study how information is encoded in sequence-to-sequence models without capacity limitations, whereas in our study, the VAE objective puts severe constraints on the capacity.

B.5. Semi-supervised learning experiments

B.5.1. Model selection

For a given dataset in a given data-regime, we want a measure of the performance of our models that abstracts away from i) hyperparameters for the VAEs, ii) hyperparameters for the downstream task classifiers, iii) subsampling of the dataset and iv) parameter initialisation of the VAEs. As is usually done by practitioners, we optimize over the hyperparameters of the VAEs and the classifiers, eliminating i) and ii) as sources of variance. We can study the robustness of the models by looking at the variance induced by the choice of the subsample and the initialisation of the parameters.

On a given dataset and in a given data-regime, for a given model, we note $F_{ij}^{H_M,H_C}$ the F1-score obtained on the test set on the subsample using seed i, the parameter initialisation using seed j, VAE hyperparameters H_M and classifier hyperparameters H_C. We use repeated stratified K-fold cross-validation [Moss et al., 2018] to compute a validation error $\widehat{F_{ij}^{H_M,H_C}}$. For all training folds, we train logistic regression classifiers with L_2 regularisation and a grid-search on $H_C \in \{0.01, 0.1, 1, 10, 100\}$. We select the best classifier hyperparameter:

$$H_C^* = \arg\max_{H_C} \widehat{F_{ij}^{H_M,H_C}}$$

Then, the best VAE hyperparameter is chosen by averaging over the $s = 3$ random seeds and picking the best classifier hyperparameter,

$$H_M^* = \arg\max_{H_M} \frac{1}{s} \sum_{i=1}^{s} \widehat{F_{ij}^{H_M,H_C^*}}$$

Having optimised the hyperparameters, we compute the test set F1-score:

$$F_{ij} = F_{ij}^{H_M^*,H_C^*}$$

B.5.2. Decomposing the variances of the scores

For a given model, dataset and data-regime, after optimisation of the hyperparameters of the VAE and the classifier, we collect several F1-scores F_{ij} which depend on the seed used to subsample the dataset i and the seed used to initialise the model parameters j. We posit a linear model with one random-effect factor, the initialisation seed, and where replicates are obtained by varying the subsampling seed:

$$F_{ij} = \mu + \alpha_j + \epsilon_{ij}$$

Assuming that α_j and ϵ_{ij} are independent random variables with null expectations, we can decompose the variance as

$$\begin{aligned}\mathrm{Var}(F_{ij}) &= \mathbb{E}[(F_{ij} - \mu)^2] \\ &= \mathbb{E}[(\alpha_i + \epsilon_{ij})^2] \\ &= \mathbb{E}[\alpha_i^2] + \mathbb{E}[\epsilon_{ij}^2] \\ &= \mathrm{Var}(\alpha_i) + \mathrm{Var}(\epsilon_{ij})\end{aligned}$$

This is the basis of the method of analysis of variance (ANOVA) and is often used to test hypotheses (for instance, that the effect $\mathbb{E}[\alpha_i]$ is significant) [Oehlert, 2010]. The two estimates of σ^2_{init} and σ^2 are usually denoted MS_T and MS_E.

In our case, we are only interested in estimating roughly what variability is due to the model initialisation and what is due to the subsampling of the dataset.

Note that we could treat the two sources of variance i and j symmetrically by adding add a term β_i, but we would need to report 3 standard deviations (that of α_j, β_i and ϵ_{ij}) to get the full picture. The most important estimate is σ_{init}. It quantifies the inherent robustness of the model to different initialisations. The effect of the subsampling is specific to the dataset, therefore, it is less relevant to our analysis.

B.5.3. What is the representation of a document?

VAEs are mostly used for generating samples but are also sometimes used as feature extractors for SSL. In the latter case, it is not clear what the representation of a datapoint is: the mean of the approximate posterior μ or the noisy samples $Z \sim \mathcal{N}(\mu, I\sigma^2)$? Kingma et al. [2014] feed noisy samples z in the classifiers but in the literature of VAEs applied to language modeling, it is more common to use μ without explanation or even mention.[2]

If we are interested purely in downstream task performance, the mean should perform best, as the samples are just noisy versions of the mean vector (it is still not completely straightforward as the noise could play a regularizing role). However, in order to evaluate what information is *effectively* transmitted to the decoder, we should use the samples. The performance of downstream task classifiers using the mean does not tell us *at all* whether the latent variable is used by the decoder to reconstruct the input. The following experiment illustrates this fact.

We train the original VAE architecture on the Yelp dataset, both with and without the *PreAE*, using the original ELBo objective ($\lambda = 0$). As expected, the KL term collapses to 0. Then, we train a classifier using the procedure explained above using 5000 examples per class. We expect that its performance will be close to random chance, regardless of whether

[2]For instance, Li et al. [2019] and Fu et al. [2019] do not mention what representation they use but their code uses the mean; Long et al. [2019] report using a concatenation of the mean and the variance vectors.

PreAE	F1		KL
	z	μ	
No	49.5	64.7	$1e^{-4}$
Yes	49.6	81.5	$2e^{-4}$

Table 5. When the KL collapses, the performances of classifiers trained on the mean μ vs on samples $z \sim \mathcal{N}(\mu, I\sigma^2)$ are very different, especially for pretrained models. z does not contain any information while μ is very predictive of the label.

samples or the mean parameter are used as inputs. However, Table 5 shows that this is not the case. Using samples, we do get random chance predictions from the classifiers, whereas using means, the performance is remarkably high (as high as 81.5 of F1 using pretraining). The reason is that the KL term never *completely* collapses to 0. Therefore, μ can be almost zero while still encoding a lot of information about its inputs. However, when the KL term is close to 0, the variance of the samples is close to 1, so no information is transmitted to the decoder. This tendency is exacerbated with the *PreAE* runs, for which the means encode remnants of the pretraining phase.

This experiment shows that it is crucial to report what representation (z or μ) is analyzed and to cautiously interpret the results. Therefore, for the purpose of analysing representations for text generation, we feed z as inputs to the classifiers.

B.5.4. Recurrent and *BoW* encoders work around max-pooling

It is counter-intuitive that *BoW-max-LSTM* improves over *LSTM-max-LSTM* (with or without *PreAE*). Indeed, taking into account word order should allow the LSTM encoder to do better inference than the *BoW* encoder, for example, by handling negation or parsing more complicated discourse structure [Pang et al., 2002].

LSTM encoders are more powerful, but it can lead them to learn undesirable behaviors. We noticed that some components of the hidden states consistently reach their maximum values at fixed positions, regardless of the inputs (i.e., for some components j^*, $\arg\max_i h_i^{j^*} \approx K$). These positions K are often early positions in the sentence. For instance, with $\lambda = 8$, $d = 16$, *LSTM-max-LSTM-PreAE* has 70 components out of 512 that are selected on 80% of the documents on the same position on Yelp (68 on the first word, 2 on the second) and 78 on Amazon (57 on the first word, 21 on the second). In other words, some components of r act like memory slots assigned to fixed positions in the sentence. This is probably achieved through counting mechanisms [Shi et al., 2016, Suzgun et al., 2019]. The decoder is also an LSTM and can count, so it can also extract the relevant components at each position to retrieve the corresponding words.

	r	Dec.	Pre.	5	50	500	5000	All
						F1$\pm^{\sigma}_{\sigma_{\text{init}}}$		
AGNews	last	LSTM	-	$59.6\pm^{5.1}_{11.9}$	$71.7\pm^{1.0}_{12.1}$	$73.6\pm^{0.1}_{11.8}$	$73.7\pm^{0.1}_{11.9}$	$73.6\pm^{-}_{5.4}$
	last	LSTM	AE	$65.8\pm^{3.3}_{3.3}$	$81.0\pm^{0.7}_{1.1}$	$82.8\pm^{0.3}_{0.6}$	$83.1\pm^{0.1}_{0.7}$	$83.4\pm^{-}_{0.3}$
	max	LSTM	-	$27.3\pm^{2.4}_{1.2}$	$30.8\pm^{3.4}_{5.4}$	$33.1\pm^{0.9}_{10.5}$	$33.8\pm^{0.4}_{8.6}$	$34.6\pm^{-}_{2.4}$
	max	LSTM	AE	$55.7\pm^{4.5}_{18.7}$	$75.1\pm^{1.3}_{2.6}$	$81.9\pm^{0.3}_{0.0}$	$82.5\pm^{0.1}_{0.4}$	$83.3\pm^{-}_{0.4}$
	max	LSTM	-	$72.7\pm^{2.0}_{5.9}$	$81.2\pm^{0.6}_{0.8}$	$82.2\pm^{0.2}_{0.8}$	$82.3\pm^{0.1}_{1.0}$	$83.1\pm^{-}_{0.3}$
	max	Uni	-	$71.6\pm^{5.5}_{0.1}$	$80.4\pm^{0.8}_{0.7}$	$81.8\pm^{0.5}_{0.5}$	$82.4\pm^{0.1}_{0.4}$	$83.9\pm^{-}_{0.3}$
	last	Uni	-	$54.8\pm^{5.2}_{57.1}$	$61.7\pm^{0.8}_{71.4}$	$62.9\pm^{0.4}_{71.0}$	$63.0\pm^{0.3}_{71.1}$	$59.3\pm^{-}_{40.9}$
	max	Uni	-	$71.8\pm^{4.5}_{1.8}$	$81.4\pm^{0.5}_{0.6}$	$82.5\pm^{0.1}_{0.5}$	$82.5\pm^{0.1}_{0.6}$	$83.1\pm^{-}_{0.5}$
	avg	LSTM	LM	$70.8\pm^{4.8}_{4.3}$	$81.2\pm^{0.9}_{1.2}$	$82.6\pm^{0.2}_{1.3}$	$82.8\pm^{0.1}_{0.9}$	$83.5\pm^{-}_{0.1}$
Amazon	last	LSTM	-	$18.9\pm^{1.7}_{0.5}$	$20.9\pm^{1.2}_{0.9}$	$22.5\pm^{0.7}_{0.7}$	$23.3\pm^{0.4}_{1.1}$	$22.9\pm^{-}_{1.5}$
	last	LSTM	AE	$20.0\pm^{2.2}_{0.9}$	$24.7\pm^{0.7}_{2.8}$	$27.2\pm^{0.4}_{3.1}$	$27.7\pm^{0.3}_{3.8}$	$28.1\pm^{-}_{1.0}$
	max	LSTM	-	$19.8\pm^{0.7}_{0.5}$	$20.4\pm^{1.1}_{0.9}$	$22.2\pm^{0.6}_{2.1}$	$23.0\pm^{0.3}_{1.9}$	$23.7\pm^{-}_{0.5}$
	max	LSTM	AE	$22.3\pm^{2.6}_{0.7}$	$30.5\pm^{0.9}_{3.0}$	$33.4\pm^{0.4}_{4.1}$	$34.1\pm^{0.3}_{4.8}$	$34.0\pm^{-}_{1.6}$
	max	LSTM	-	$21.0\pm^{2.6}_{1.1}$	$34.6\pm^{0.7}_{1.1}$	$38.3\pm^{0.4}_{1.0}$	$39.0\pm^{0.1}_{0.6}$	$38.9\pm^{-}_{0.7}$
	max	Uni	-	$21.8\pm^{3.1}_{1.6}$	$32.8\pm^{0.8}_{1.7}$	$36.9\pm^{0.4}_{0.9}$	$38.0\pm^{0.2}_{0.6}$	$38.2\pm^{-}_{0.5}$
	last	Uni	-	$24.0\pm^{3.0}_{1.0}$	$31.2\pm^{0.6}_{1.4}$	$35.1\pm^{0.4}_{2.2}$	$36.1\pm^{0.2}_{2.4}$	$36.8\pm^{-}_{0.9}$
	max	Uni	-	$25.4\pm^{3.2}_{0.2}$	$32.8\pm^{1.0}_{1.3}$	$36.1\pm^{0.4}_{0.7}$	$36.9\pm^{0.2}_{0.8}$	$37.9\pm^{-}_{0.2}$
	avg	LSTM	LM	$21.8\pm^{3.8}_{0.6}$	$35.3\pm^{0.8}_{0.4}$	$40.2\pm^{0.4}_{0.4}$	$41.1\pm^{0.2}_{0.4}$	$40.0\pm^{-}_{0.4}$
Yahoo	last	LSTM	-	$10.9\pm^{0.9}_{0.5}$	$12.1\pm^{0.6}_{0.6}$	$13.9\pm^{0.4}_{2.1}$	$14.1\pm^{0.2}_{2.8}$	$14.9\pm^{-}_{1.0}$
	last	LSTM	AE	$20.7\pm^{0.7}_{0.5}$	$32.2\pm^{0.8}_{0.6}$	$36.1\pm^{0.2}_{0.1}$	$36.7\pm^{0.1}_{0.5}$	$37.2\pm^{-}_{0.7}$
	max	LSTM	-	$9.9\pm^{1.0}_{1.3}$	$13.0\pm^{0.6}_{2.1}$	$14.6\pm^{0.3}_{2.8}$	$14.9\pm^{0.1}_{3.1}$	$15.7\pm^{-}_{0.5}$
	max	LSTM	AE	$20.8\pm^{1.3}_{2.3}$	$31.3\pm^{0.7}_{1.4}$	$35.6\pm^{0.3}_{1.2}$	$36.3\pm^{0.1}_{1.1}$	$36.6\pm^{-}_{0.7}$
	max	LSTM	-	$23.4\pm^{2.1}_{2.9}$	$36.7\pm^{1.1}_{0.5}$	$41.1\pm^{0.2}_{0.8}$	$41.6\pm^{0.1}_{0.9}$	$42.6\pm^{-}_{0.2}$
	max	Uni	-	$24.9\pm^{1.3}_{2.2}$	$33.2\pm^{0.7}_{3.6}$	$37.3\pm^{0.1}_{3.1}$	$37.9\pm^{0.1}_{3.1}$	$38.9\pm^{-}_{1.7}$
	last	Uni	-	$24.5\pm^{3.8}_{1.7}$	$30.8\pm^{1.7}_{0.6}$	$34.4\pm^{0.3}_{5.0}$	$35.1\pm^{0.1}_{4.7}$	$37.1\pm^{-}_{2.3}$
	max	Uni	-	$24.1\pm^{2.9}_{2.7}$	$35.0\pm^{0.9}_{1.2}$	$39.1\pm^{0.1}_{1.8}$	$39.5\pm^{0.1}_{1.7}$	$40.1\pm^{-}_{0.7}$
	avg	LSTM	LM	$21.9\pm^{2.3}_{1.3}$	$36.1\pm^{0.8}_{0.7}$	$39.9\pm^{0.2}_{0.6}$	$40.4\pm^{0.1}_{0.4}$	$41.7\pm^{-}_{0.3}$
Yelp	last	LSTM	-	$49.9\pm^{4.5}_{2.7}$	$55.6\pm^{2.3}_{2.9}$	$57.9\pm^{1.1}_{2.5}$	$59.5\pm^{0.2}_{2.7}$	$61.9\pm^{-}_{2.5}$
	last	LSTM	AE	$59.3\pm^{5.4}_{2.9}$	$80.0\pm^{1.3}_{3.0}$	$82.7\pm^{1.0}_{0.9}$	$83.3\pm^{0.1}_{2.3}$	$67.9\pm^{-}_{0.1}$
	max	LSTM	-	$61.6\pm^{8.2}_{8.8}$	$71.4\pm^{2.3}_{6.3}$	$76.0\pm^{0.2}_{2.3}$	$76.5\pm^{0.1}_{2.0}$	$78.0\pm^{-}_{1.7}$
	max	LSTM	AE	$59.9\pm^{10.4}_{7.9}$	$78.7\pm^{2.4}_{1.5}$	$82.9\pm^{0.3}_{2.7}$	$83.3\pm^{0.1}_{2.7}$	$84.1\pm^{-}_{0.7}$
	max	LSTM	-	$67.1\pm^{10.1}_{15.7}$	$79.3\pm^{2.8}_{4.5}$	$83.4\pm^{0.3}_{0.9}$	$83.9\pm^{0.1}_{0.9}$	$85.0\pm^{-}_{0.2}$
	max	Uni	-	$62.3\pm^{4.6}_{3.8}$	$76.7\pm^{1.7}_{3.6}$	$80.4\pm^{0.2}_{3.2}$	$80.9\pm^{0.1}_{3.1}$	$83.1\pm^{-}_{0.5}$
	last	Uni	-	$65.0\pm^{8.0}_{4.4}$	$74.1\pm^{2.0}_{1.4}$	$78.5\pm^{0.3}_{3.0}$	$79.1\pm^{0.1}_{3.1}$	$81.6\pm^{-}_{0.5}$
	max	Uni	-	$59.9\pm^{7.2}_{3.7}$	$77.3\pm^{1.2}_{0.9}$	$81.1\pm^{0.3}_{0.5}$	$81.5\pm^{0.1}_{0.5}$	$83.3\pm^{-}_{0.4}$
	avg	LSTM	LM	$63.6\pm^{7.4}_{5.4}$	$81.0\pm^{1.6}_{2.3}$	$83.2\pm^{0.7}_{0.8}$	$83.8\pm^{0.1}_{0.8}$	$84.4\pm^{-}_{0.5}$

Table 6. Using *BoW* encoders, *Uni* decoders or *PreLM* pretraining, the learned representations are more predictive of the labels (sentiment or topic) of the documents.

For *BoW* encoders, it is less clear. It is possible that on some datasets, capitalized words could take especially high values on some components, in order to be consistently represented after max pooling. However, we have not explored the issue further.

Dataset	F1(All)	F1(3)	Ratio
AGNews	89.0	71.9	0.808
Amazon	48.9	29.7	0.607
Yahoo	63.0	19.1	0.303
Yelp	96.5	82.4	0.854

Table 7. Performance of bag-of-word classifiers when using all words as features versus only the first three words. Ratios of performance vary a lot across datasets.

B.6. Qualitative analysis

For our qualitative analysis, we take a look at the reconstruction samples (which were also used in Section 7.6). We focus on the *PreUni* models which lower memorization the most with the *LSTM-max* and *BoW-max* encoders, and compare them to the two best baselines. We use only one seed and one z sample per model and per source sentence, but use two decoding strategies (beam search and greedy decoding).

In general, and for the reasons explained above, the rate $\lambda = 8$ is chosen too small to recover exactly the source. Indeed, this rate is an order of magnitude less than the negative log-likelihood of *LSTM-LM* baselines: above 80 for all datasets except on Yelp where it is around 34.60 nats (cf. Table 3). Since the NLL is an upper-bound on the entropy of the data, it gives a crude over estimate of the information content of the average document. On Yelp, where the NLL is much smaller (around 34.60 nats), we hope to obtain good paraphrases for simple and frequent sentences. On the other datasets, we can not hope to reconstruct the sentences correctly but merely to control the generation by producing sentences which have the same labels as the source sentences. For this reason, we cherry-pick source sentences that look quite generic, because they are more probable and therefore, should be easy to reconstruct correctly.

Results are presented in Tables 8, 9, 10 and 11. Overall, we do observe less memorization of the first words and more correct sentiment or topic. Between our two models, on Amazon and Yelp (sentiment labels), it seems that *LSTM-max-LSTM-PreUni* might perform better than *BoW-max-LSTM-PreUni* because of its ability to handle negation, probably thanks to the recurrent encoder. It also seem more on topic on AGNews. On Yelp, it is able to paraphrase generic, small sentences. Therefore, we recommend this model as a future baseline.

We have already seen that it is very hard to classify sentences based on their first three words on the Amazon and Yahoo dataset, and that the baseline methods will learn representations that are not predictive of labels. However, the Yahoo dataset is especially challenging and our methods also struggle on it. We hypothesize that it is because only a few words per sentences are correlated with the labels. Indeed, there are many sentences of the form "what do you think about X?" or "what is the difference between X and Y?" where

only X and Y are correlated with the label and moreover often out-of-vocabulary. There might be no benefits for the model to diminish the reconstruction loss in priority on these words. By comparison, the other "hard" dataset (Amazon) is "easier", because the sentiment is often indicated by frequent adjectives like "horrible" or "good" and other frequent verbs.

Model/“source"	Dec.	Sample
source		Michael Owen heads England ’s winner in the World Cup qualifier against Azerbaijan .
L-last-L-PreAE	beam	Michael Owen will be sidelined for the rest of the season with a knee injury .
L-last-L-PreAE	greedy	Michael Owen has been charged with a rib injury and a new team for the first time in the last two weeks .
L-max-L-PreAE	beam	American Bode Miller won the World Cup super @-@ G with a 6 @-@ 3 victory over the United States in the World Cup of Hockey .
L-max-L-PreAE	greedy	American Bode Miller won the World Cup of Hockey on Sunday , beating the United States by a record @-@ setting victory over the United States .
B-max-L-PreUni	beam	England coach Sven @-@ Goran Eriksson says he will not be able to win the World Cup qualifier against Wales .
B-max-L-PreUni	greedy	England captain David Beckham has been named the England captain for the 2006 World Cup qualifiers against Wales .
L-max-L-PreUni	beam	England coach Sven @-@ Goran Eriksson says he will be fit for the World Cup qualifier against Wales next month .
L-max-L-PreUni	greedy	England captain David Beckham has been named the first World Cup qualifier in the World Cup qualifier against Wales .
source		New Athlon 64 processors will compete with Intel ’s Pentium 4 Extreme Edition .
L-last-L-PreAE	beam	IBM ’ s dual @-@ core Opteron processor will be available in the next three years .
L-last-L-PreAE	greedy	A new chipset for mobile phones will be available in the next three years .
L-max-L-PreAE	beam	New version of Windows Server 2003 .
L-max-L-PreAE	greedy	New version of the Linux operating system is designed to integrate Linux and Linux .
B-max-L-PreUni	beam	Hewlett @-@ Packard Co . , the world ’s largest computer maker , has unveiled a new version of its iPod digital music player , the company said .
B-max-L-PreUni	greedy	Hewlett @-@ Packard Co . , the world ’s largest maker of digital music player , on Tuesday unveiled a new version of its popular PlayStation 2 game console , which will be available in the next few years .
L-max-L-PreUni	beam	Intel has unveiled a new version of its Pentium 4 Extreme Edition processor , which will be available for the first time .
L-max-L-PreUni	greedy	Intel has unveiled a new version of its Pentium M processor , which is designed to help the company ’s new processor @-@ based processors .
source		Nortel said it expects revenue for the third quarter to fall short of expectations .
L-last-L-PreAE	beam	Coca @-@ Cola Co .
L-last-L-PreAE	greedy	research) is expected to announce a new deal with the company to buy the company .
L-max-L-PreAE	beam	Nortel Networks Corp.
L-max-L-PreAE	greedy	General Electric Co. said on Thursday it will buy the company for $ 500 million in cash and stock .
B-max-L-PreUni	beam	Ford Motor Co . , the world ’s largest maker of photographic film , said on Thursday it expects to cut its full @-@ year earnings forecast , citing strong demand for its flagship database products .
B-max-L-PreUni	greedy	Coca @-@ Cola Co. on Thursday said third @-@ quarter earnings rose 29 percent , helped by strong sales of its soft drinks and business software .
L-max-L-PreUni	beam	Nortel Networks Corp. , the world ’s largest maker of equipment , said on Thursday that its third @-@ quarter profit rose 12 percent , helped by a rebound in the value of its assets .
L-max-L-PreUni	greedy	Shares of Nortel Networks Corp. fell nearly 8 percent on Thursday after the company said it expects its earnings for the third quarter , citing a decline in its third @-@ quarter earnings .

Table 8. Cherry-picked AGNews samples. L=LSTM; B=BoW. Baselines are the first two models, our models are the two last. In the first example, the first baseline copies “Michael Owen” and complete with generic suffixes; the second baseline is about Hockey instead of soccer. Our baselines do not copy the beginning while correctly identifying the topic of England and the World Cup’s qualifier. Similar comments can be made on the two other examples.

Model/“source"	Dec.	Sample
source		I loved the book , but was a little bit UNK with the ending .
L-last-L-PreAE	beam	I enjoyed the book , but the story line was not as good as the first one .
L-last-L-PreAE	greedy	I enjoyed the book and the story line . I was very disappointed in the book .
L-max-L-PreAE	beam	I was very disappointed in the quality of the book , and the content of the book is very poor .
L-max-L-PreAE	greedy	I was disappointed in the quality of the book , but the book is not as good as the original .
B-max-L-PreUni	beam	I liked the story and the story line . It was a little slow at times but overall a good read .
B-max-L-PreUni	greedy	I liked the story and the story line . It was a little slow but the ending was a little predictable .
L-max-L-PreUni	beam	The book was a little slow , but the story line was good . I enjoyed it .
L-max-L-PreUni	greedy	The book was a little slow and the story line was very good . I was very disappointed .
source		This movie wasn ’t as good as the original but I still enjoy watching it .
L-last-L-PreAE	beam	This movie was a little slow at times , but it was a good movie .
L-last-L-PreAE	greedy	This movie was a little slow and the plot was not good . I would not recommend it to anyone .
L-max-L-PreAE	beam	This movie was not as good as I thought it would be . I was very disappointed .
L-max-L-PreAE	greedy	The movie was not as good as the first one . I was disappointed in the quality of the movie .
B-max-L-PreUni	beam	Not as good as I thought it would be . I wouldn ’t watch it again .
B-max-L-PreUni	greedy	I didn ’t like this movie . I thought it was going to be a good movie but I wouldn ’t watch it again .
L-max-L-PreUni	beam	I was expecting a little more from this movie . It was a little slow and boring .
L-max-L-PreUni	greedy	The movie was good , but the acting was not very good . I was expecting a little more from the movie .
source		This movie is horrible . The story , the acting , the directing . Just horrible .
L-last-L-PreAE	beam	This is a great movie . I love it . It is a great family movie .
L-last-L-PreAE	greedy	This movie is great . It is a great movie and I love it .
L-max-L-PreAE	beam	This is the worst movie I have ever seen . It was not worth the time to watch .
L-max-L-PreAE	greedy	This was a good movie . The acting was good , but the story line was not very good .
B-max-L-PreUni	beam	This movie is not worth the money . The acting is poor and the acting is poor .
B-max-L-PreUni	greedy	The movie is very poor , the acting is poor . The acting is poor .
L-max-L-PreUni	beam	This movie is a waste of time and money . It was a waste of time and money .
L-max-L-PreUni	greedy	This movie is a waste of time . It was a waste of time and money .
source		This book is very bad and does not give a real idea of the sport of UNK
L-last-L-PreAE	beam	this is a great book for those who want to learn a little more about the history of the history of the history
L-last-L-PreAE	greedy	this book is a great book for the price , but the book is a little too short for my taste
L-max-L-PreAE	beam	This book is a must have for anyone who is interested in the history of the Catholic Church
L-max-L-PreAE	greedy	This book is a must have for anyone who is interested in the field of the world of the New Testament
B-max-L-PreUni	beam	This is a good book , but it does not have a lot of information in it .
B-max-L-PreUni	greedy	This book is a good book for the beginner , but it does not have a lot of information in it .
L-max-L-PreUni	beam	This is a good book , but it does not have enough information on how to build your own .
L-max-L-PreUni	greedy	This is a good book , but the book is not a good book for the money .

Table 9. Cherry-picked Amazon samples. L=LSTM; B=BoW. The first two examples are moderately positive reviews. L-max-L-PreUNI recreate roughly the same sentiment overall with different beginnings of sentences. On the third example, our models do copy the beginning of the source sentence but do not make mistakes on the sentiment as the baselines do. On the last, all models fail to capture the very negative sentiment, but our models at least moderate the positivity.

Model/“source"	Dec.	Sample
source		What is your favorite book ?
LSTM-last-LSTM-PreAE	beam	what is a good name for a girl ?
LSTM-last-LSTM-PreAE	greedy	what is a good name for a girl ?
LSTM-max-LSTM-PreAE	beam	What is your favorite book ?
LSTM-max-LSTM-PreAE	greedy	What is your favorite book ?
BoW-max-LSTM-PreUni	beam	What is your favorite sport ?
BoW-max-LSTM-PreUni	greedy	What is your favorite movie ?
LSTM-max-LSTM-PreUni	beam	What is your favorite book ?
LSTM-max-LSTM-PreUni	greedy	What is your favorite Christmas movie ?
source		how can i get money ?
LSTM-last-LSTM-PreAE	beam	i want to know ?
LSTM-last-LSTM-PreAE	greedy	i want to know ?
LSTM-max-LSTM-PreAE	beam	how do i get money ?
LSTM-max-LSTM-PreAE	greedy	do you have any money ?
BoW-max-LSTM-PreUni	beam	where can i go to watch free online ?
BoW-max-LSTM-PreUni	greedy	where can i get free online games ?
LSTM-max-LSTM-PreUni	beam	where can i get a job ?
LSTM-max-LSTM-PreUni	greedy	how do i get a job in USA ?
source		What countries have nuclear weapons ?
LSTM-last-LSTM-PreAE	beam	What are the pros and cons ?
LSTM-last-LSTM-PreAE	greedy	What are the pros and cons of smoking ?
LSTM-max-LSTM-PreAE	beam	which countries in the world ?
LSTM-max-LSTM-PreAE	greedy	which country has the highest world cup ?
BoW-max-LSTM-PreUni	beam	how many countries are there in the world ?
BoW-max-LSTM-PreUni	greedy	what are the countries that will be able to be the most effective government in the world ?
LSTM-max-LSTM-PreUni	beam	Why are the colors of the Earth ?
LSTM-max-LSTM-PreUni	greedy	what are the three different countries in the U.S. ?
source		how to print all webpage content ?
LSTM-last-LSTM-PreAE	beam	how to create a website ?
LSTM-last-LSTM-PreAE	greedy	how to find a website ?
LSTM-max-LSTM-PreAE	beam	how can i learn english language ?
LSTM-max-LSTM-PreAE	greedy	how can i watch free online online ?
BoW-max-LSTM-PreUni	beam	how do you get a free copy of the internet ?
BoW-max-LSTM-PreUni	greedy	how to get the free internet explorer ?
LSTM-max-LSTM-PreUni	beam	how do i get a copy of my computer in the internet ?
LSTM-max-LSTM-PreUni	greedy	how do i get the power to open a computer in the internet ?

Table 10. Cherry-picked Yahoo samples. There isn't a model that clearly stands out, but we can rule out LSTM-last-LSTM-PreAE. This dataset is more difficult (see main text).

Model/“source"	Dec.	Sample
source		amazing place .
LSTM-last-LSTM-PreAE	beam	amazing customer service .
LSTM-last-LSTM-PreAE	greedy	amazing customer service .
LSTM-max-LSTM-PreAE	beam	amazing food .
LSTM-max-LSTM-PreAE	greedy	amazing food .
BoW-max-LSTM-PreUni	beam	this place is amazing .
BoW-max-LSTM-PreUni	greedy	this place is amazing .
LSTM-max-LSTM-PreUni	beam	this place is amazing .
LSTM-max-LSTM-PreUni	greedy	this place is amazing .
source		definitely going back soon !
LSTM-last-LSTM-PreAE	beam	definitely coming back !
LSTM-last-LSTM-PreAE	greedy	definitely recommend to anyone !
LSTM-max-LSTM-PreAE	beam	definitely coming back again !
LSTM-max-LSTM-PreAE	greedy	highly recommend them to anyone !
BoW-max-LSTM-PreUni	beam	i will definitely be back !
BoW-max-LSTM-PreUni	greedy	i will definitely be back !
LSTM-max-LSTM-PreUni	beam	i will be back !
LSTM-max-LSTM-PreUni	greedy	i will be back !
source		not worth the risk .
LSTM-last-LSTM-PreAE	beam	not the best .
LSTM-last-LSTM-PreAE	greedy	not the best .
LSTM-max-LSTM-PreAE	beam	not worth the money .
LSTM-max-LSTM-PreAE	greedy	not worth the money .
BoW-max-LSTM-PreUni	beam	worth the wait .
BoW-max-LSTM-PreUni	greedy	it was worth the wait .
LSTM-max-LSTM-PreUni	beam	not worth the hassle .
LSTM-max-LSTM-PreUni	greedy	it ’s not worth the money .
source		overall , a huge disappointment .
LSTM-last-LSTM-PreAE	beam	pizza was good too .
LSTM-last-LSTM-PreAE	greedy	pizza was good too .
LSTM-max-LSTM-PreAE	beam	ok , nothing special .
LSTM-max-LSTM-PreAE	greedy	nothing special , but the food was bland .
BoW-max-LSTM-PreUni	beam	wow .
BoW-max-LSTM-PreUni	greedy	great experience .
LSTM-max-LSTM-PreUni	beam	what a disappointment .
LSTM-max-LSTM-PreUni	greedy	what a disappointment .

Table 11. Cherry-picked Yelp samples. On small and typical sentences, our last variant LSTM-max-LSTM-PreUni can produce paraphrases. On the other hand, BoW-max-LSTM-PreUni fails on the two negative examples, probably because it lacks the ability to deal with negation. The baseline models also fail to capture the sentiment on the last example, and copy the beginning on the first three examples.