Deep Generative Language Models

DGM4NLP

Miguel Rios
University of Amsterdam

May 2, 2019

Outline

Recap Generative Models of Word Representation

Discriminative embedding models
word2vec

In the event of a chemical spill, most children know they should **evacuate** *as advised by people in charge.*

Place words in $\mathbb{R}^d$ as to answer questions like

"Have I seen this word in this context?"

Recap Generative Models of Word Representation

Discriminative embedding models

word2vec

In the event of a chemical spill, most children know they should **evacuate** *as advised by people in charge.*

Place words in $\mathbb{R}^d$ as to answer questions like

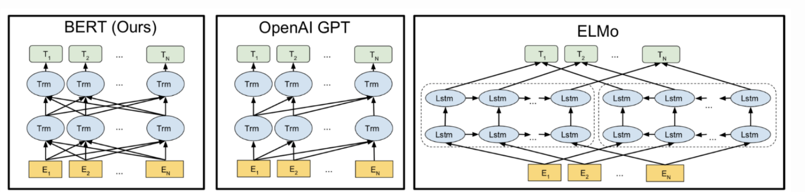
"Have I seen this word in this context?"

Fit a binary classifier

- positive examples
- negative examples

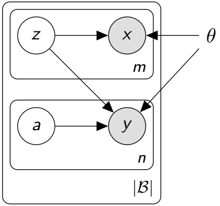
Recap Generative Models of Word Representation

- The models processes a sentence and outputs a word representation:



Recap Generative Models of Word Representation

quickly evacuate the area / deje el lugar rápidamente



X

Z

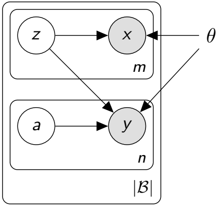
A

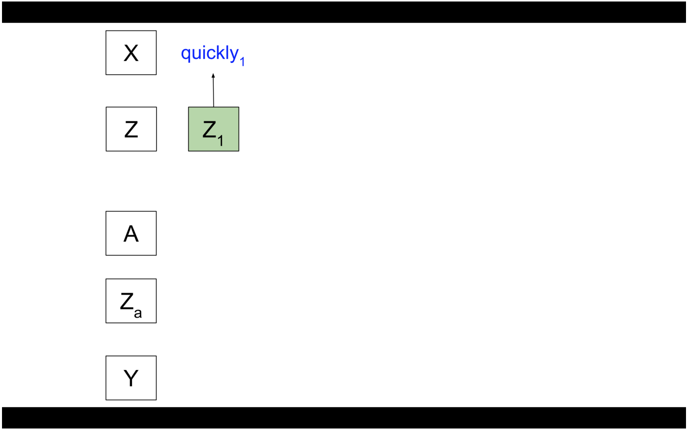
Z_a

Y

Recap Generative Models of Word Representation

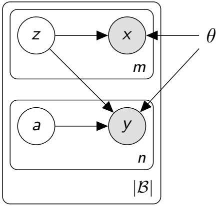
quickly evacuate the area / deje el lugar rápidamente

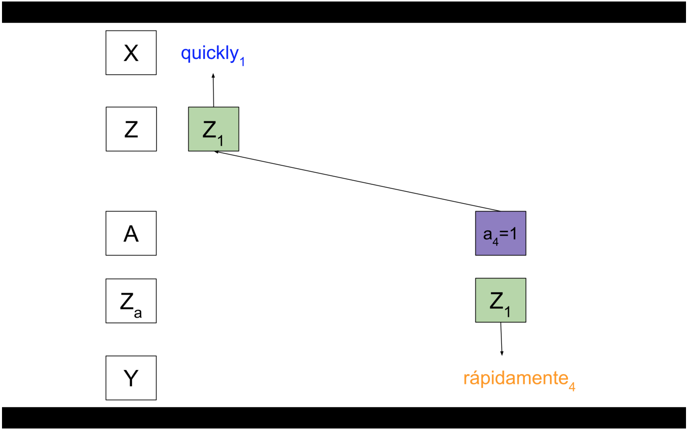




Recap Generative Models of Word Representation

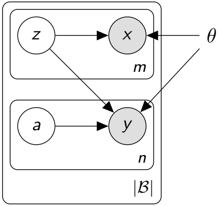
quickly evacuate the area / deje el lugar rápidamente

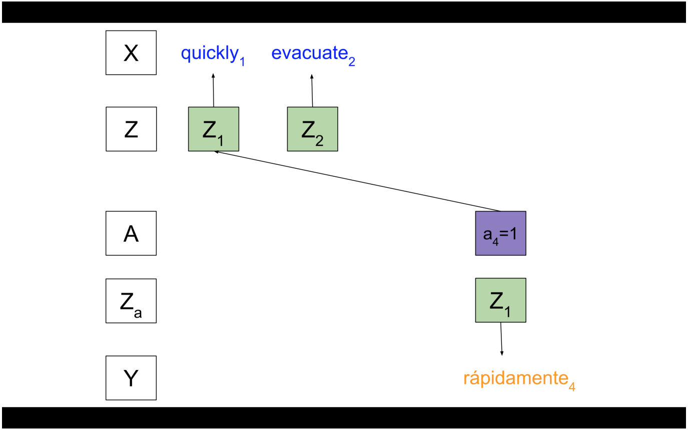




Recap Generative Models of Word Representation

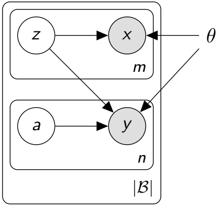
quickly evacuate the area / deje el lugar rápidamente

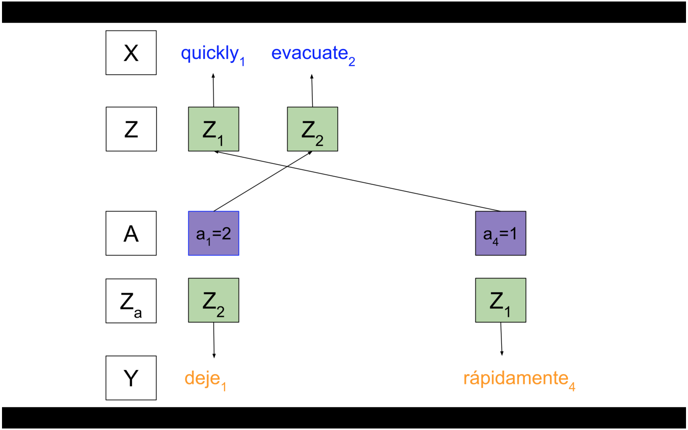




Recap Generative Models of Word Representation

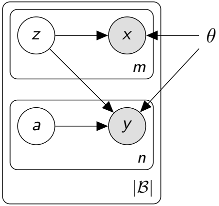
quickly evacuate the area / deje el lugar rápidamente

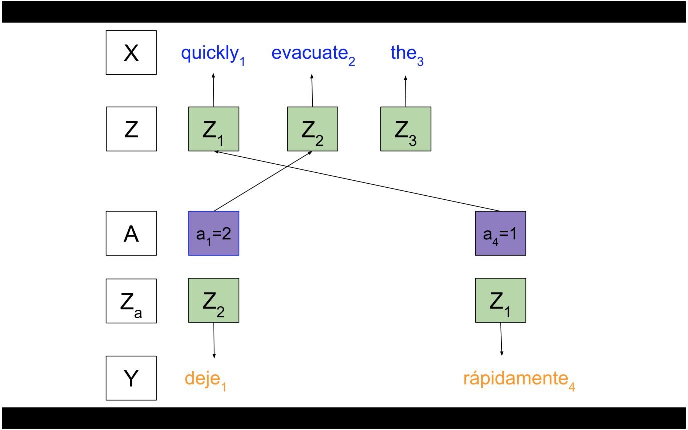




Recap Generative Models of Word Representation

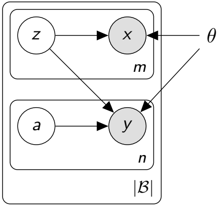
quickly evacuate the area / deje el lugar rápidamente

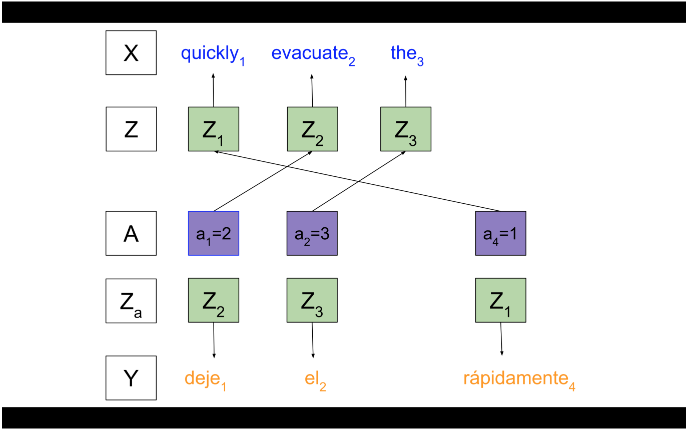




Recap Generative Models of Word Representation

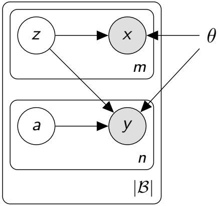
quickly evacuate the area / deje el lugar rápidamente

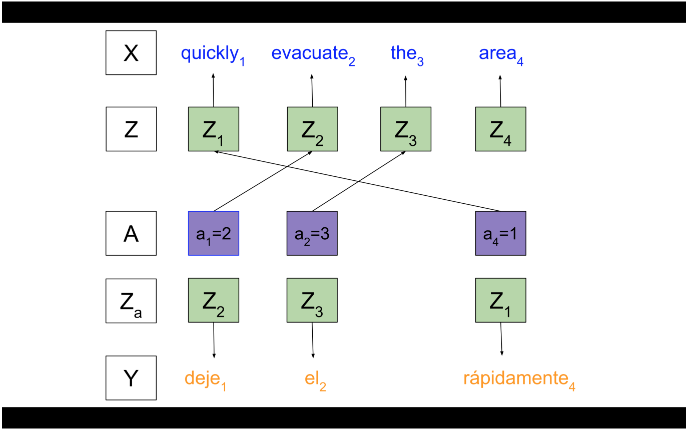




Recap Generative Models of Word Representation

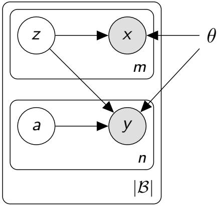
quickly evacuate the area / deje el lugar rápidamente

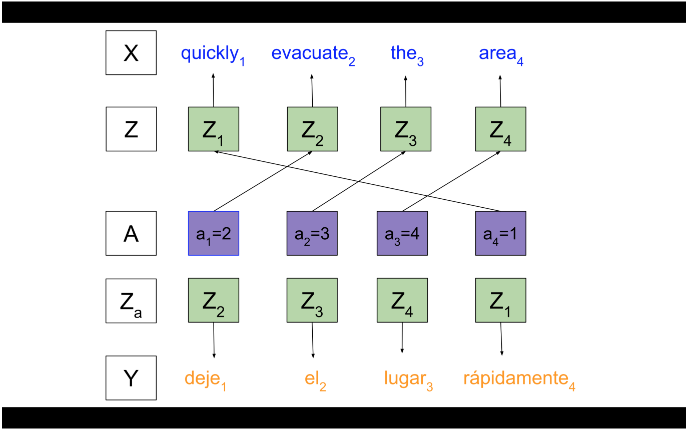




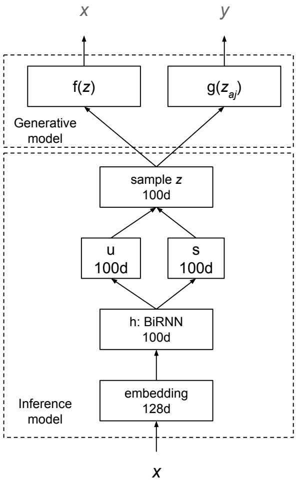
Recap Generative Models of Word Representation

quickly evacuate the area / deje el lugar rápidamente





Recap Generative Models of Word Representation



Introduction

- you know nothing, jon x

Introduction

- you know nothing, jon x
- ground control to major x

Introduction

- you know nothing, jon x
- ground control to major x
- the x

Introduction

- the quick brown x

Introduction

- the quick brown x
- the quick brown fox x

Introduction

- the quick brown x
- the quick brown fox x
- the quick brown fox jumps x

Introduction

- the quick brown x
- the quick brown fox x
- the quick brown fox jumps x
- the quick brown fox jumps over x

Introduction

- the quick brown x
- the quick brown fox x
- the quick brown fox jumps x
- the quick brown fox jumps over x
- the quick brown fox jumps over the x

Introduction

- the quick brown x
- the quick brown fox x
- the quick brown fox jumps x
- the quick brown fox jumps over x
- the quick brown fox jumps over the x
- the quick brown fox jumps over the lazy x

Introduction

- the quick brown x
- the quick brown fox x
- the quick brown fox jumps x
- the quick brown fox jumps over x
- the quick brown fox jumps over the x
- the quick brown fox jumps over the lazy x
- the quick brown fox jumps over the lazy dog

Definition

- Language models give us the probability of a sentence;

Definition

- Language models give us the probability of a sentence;
- At a time step, they assign a probability to the next word.

Applications

- Very useful on different tasks:

Applications

- Very useful on different tasks:
- Speech recognition;

Applications

- Very useful on different tasks:
- Speech recognition;
- Spelling correction;

Applications

- Very useful on different tasks:
- Speech recognition;
- Spelling correction;
- Machine translation;

Applications

- Very useful on different tasks:
- Speech recognition;
- Spelling correction;
- Machine translation;
- LMs are useful in almost any tasks that deals with generating language.

Language Models

- N-gram based LMs;

Language Models

- N-gram based LMs;
- Log-linear LMs;

Language Models

- N-gram based LMs;
- Log-linear LMs;
- Neural LMs.

N-gram LM

- x is a sequence of words

N-gram LM

- x is a sequence of words
- $x = x_1, x_2, x_3, x_4, x_5$
  $=$ you, know, nothing, jon, snow

N-gram LM

- To compute the probability of a sentence

$$p(x) = p\left(x_1, x_2, \ldots, x_n\right) \tag{1}$$

[Jelinek and Mercer, 1980, Goodman, 2001]

N-gram LM

- To compute the probability of a sentence

$$p(x) = p\left(x_1, x_2, \ldots, x_n\right) \tag{1}$$

- We apply the chain rule:

$$p(x) = \prod_i p\left(x_i | x_1, \ldots, x_{i-1}\right) \tag{2}$$

[Jelinek and Mercer, 1980, Goodman, 2001]

N-gram LM

- To compute the probability of a sentence

$$p(x) = p\left(x_1, x_2, \ldots, x_n\right) \tag{1}$$

- We apply the chain rule:

$$p(x) = \prod_i p\left(x_i | x_1, \ldots, x_{i-1}\right) \tag{2}$$

- We limit the history with a Markov order:
  $p\left(x_i | x_1, \ldots, x_{i-1}\right) \simeq p\left(x_i | x_{i-4}, x_{i-3}, x_{i-2}, x_{i-1}\right)$

[Jelinek and Mercer, 1980, Goodman, 2001]

N-gram LM

- Chain rule:

$$p(x) = \prod_i p\left(x_i | x_1, \ldots, x_{i-1}\right) \tag{3}$$

N-gram LM

- Chain rule:

$$p(x) = \prod_i p\left(x_i | x_1, \ldots, x_{i-1}\right) \tag{3}$$

-
$$\begin{aligned} P(x) =& P(\text{"you know nothing jon snow"}) \\ =& P(\text{"you"}) \cdot \\ & P(\text{"know"} \mid \text{"you"}) \cdot \\ & P(\text{"nothing"} \mid \text{"you know"}) \cdot \\ & P(\text{"jon"} \mid \text{"you know nothing"}) \cdot \\ & P(\text{"snow"} \mid \text{"you know nothing jon"}). \end{aligned}$$

N-gram LM

- We make a Markov assumption of conditional independence:

$$p\left(x_i \middle| x_1, \ldots, x_{i-1}\right) \simeq p\left(x_i \middle| x_{i-1}\right) \tag{4}$$

N-gram LM

- We make a Markov assumption of conditional independence:

$$p\left(x_i \middle| x_1, \ldots, x_{i-1}\right) \simeq p\left(x_i \middle| x_{i-1}\right) \tag{4}$$

-

$$\begin{aligned} P(x) &= P(\text{"you know nothing jon snow"}) \\ &= P(\text{"you know"}) \cdot P(\text{"know nothing"}) \cdot P(\text{"nothing jon"}) \cdot P(\text{"jon snov} \end{aligned}$$

N-gram LM

- We make the Markov assumption:
  $p\left(x_i \middle| x_1, \ldots, x_{i-1}\right) \simeq p\left(x_i \middle| x_{i-1}\right)$

N-gram LM

- We make the Markov assumption: $p\left(x_i | x_1, \ldots, x_{i-1}\right) \simeq p\left(x_i | x_{i-1}\right)$
- If we do not observe the bigram $p(x_i|x_{i-1})$ is 0 and the probability of a sentence will be 0.

N-gram LM

- We make the Markov assumption: $p\left(x_i|x_1,\ldots,x_{i-1}\right) \simeq p\left(x_i|x_{i-1}\right)$
- If we do not observe the bigram $p(x_i|x_{i-1})$ is 0 and the probability of a sentence will be 0.
- MLE

$$p_{\mathrm{MLE}}\left(x_i|x_{i-1}\right) = \frac{\text{count}\left(x_{i-1},x_i\right)}{\text{count}\left(x_{i-1}\right)} \tag{5}$$

N-gram LM

- We make the Markov assumption: $p\left(x_i|x_1,\ldots,x_{i-1}\right) \simeq p\left(x_i|x_{i-1}\right)$
- If we do not observe the bigram $p(x_i|x_{i-1})$ is 0 and the probability of a sentence will be 0.
- MLE

$$p_{\mathrm{MLE}}\left(x_i|x_{i-1}\right) = \frac{\text{count}\left(x_{i-1}, x_i\right)}{\text{count}\left(x_{i-1}\right)} \tag{5}$$

- Laplace smoothing:

$$p_{\mathrm{add1}}\left(x_i|x_{i-1}\right) = \frac{\text{count}\left(x_{i-1}, x_i\right) + 1}{\text{count}\left(x_{i-1}\right) + V} \tag{6}$$

Log-linear LM

- $$p(y|x) = \frac{\exp \boldsymbol{w} \cdot \phi(x, y)}{\sum_{y' \in V_y} \exp w \cdot \phi\left(x, y'\right)} \tag{7}$$

Log-linear LM

- $$p(y|x) = \frac{\exp \boldsymbol{w} \cdot \phi(x, y)}{\sum_{y' \in V_y} \exp w \cdot \phi\left(x, y'\right)} \tag{7}$$
- y is the next word and V_y is the vocabulary;

Log-linear LM

- $$p(y|x) = \frac{\exp \boldsymbol{w} \cdot \phi(x, y)}{\sum_{y' \in V_y} \exp w \cdot \phi(x, y')} \tag{7}$$
- y is the next word and V_y is the vocabulary;
- x is the history;

Log-linear LM

- $$p(y|x) = \frac{\exp \boldsymbol{w} \cdot \phi(x, y)}{\sum_{y' \in V_y} \exp w \cdot \phi\left(x, y'\right)} \tag{7}$$
- y is the next word and V_y is the vocabulary;
- x is the history;
- ϕ is a feature function that returns an n-dimensional vector;

Log-linear LM

- $$p(y|x) = \frac{\exp \boldsymbol{w} \cdot \phi(x, y)}{\sum_{y' \in V_y} \exp w \cdot \phi(x, y')} \tag{7}$$
- y is the next word and V_y is the vocabulary;
- x is the history;
- ϕ is a feature function that returns an n-dimensional vector;
- w are the model parameters.

Log-linear LM

- n-gram features x_{j-1} = the and x_j = puppy.

Log-linear LM

- n-gram features $x_{j-1} = \text{the}$ and $x_j = \text{puppy}$.
- gappy n-gram features $x_{j-2} = \text{the}$ and $x_j = \text{puppy}$.

Log-linear LM

- n-gram features x_{j-1} = the and x_j = puppy.
- gappy n-gram features x_{j-2} = the and x_j = puppy.
- class features: x_j belongs to class ABC;

Log-linear LM

- n-gram features $x_{j-1} =$ the and $x_j =$ puppy.
- gappy n-gram features $x_{j-2} =$ the and $x_j =$ puppy.
- class features: x_j belongs to class ABC;
- gazetteer features: x_j is a place name;

Log-linear LM

- With features of words and histories we can share statistical weight

Log-linear LM

- With features of words and histories we can share statistical weight
- With n-grams, there is no sharing at all

Log-linear LM

- With features of words and histories we can share statistical weight
- With n-grams, there is no sharing at all
- We also get smoothing for free

Log-linear LM

- With features of words and histories we can share statistical weight
- With n-grams, there is no sharing at all
- We also get smoothing for free
- We can add arbitrary features

Log-linear LM

- With features of words and histories we can share statistical weight
- With n-grams, there is no sharing at all
- We also get smoothing for free
- We can add arbitrary features
- We use Stochastic Gradient Descent (SGD)

Neural LM

- n-gram language models have proven to be effective in various tasks

Neural LM

- n-gram language models have proven to be effective in various tasks
- log-linear models allow us to share weights through features

Neural LM

- n-gram language models have proven to be effective in various tasks
- log-linear models allow us to share weights through features
- maybe our history is still too limited, e.g. n-1 words

Neural LM

- n-gram language models have proven to be effective in various tasks
- log-linear models allow us to share weights through features
- maybe our history is still too limited, e.g. n-1 words
- we need to find useful features

Feed-forward NLM

- With NN we can exploit distributed representations to allow for statistical weight sharing.

[Bengio et al., 2003]

Feed-forward NLM

- With NN we can exploit distributed representations to allow for statistical weight sharing.
- How does it work:

[Bengio et al., 2003]

Feed-forward NLM

- With NN we can exploit distributed representations to allow for statistical weight sharing.
- How does it work:
    1. each word is mapped to an embedding: an m-dimensional feature vector;

[Bengio et al., 2003]

Feed-forward NLM

- With NN we can exploit distributed representations to allow for statistical weight sharing.
- How does it work:
  1. each word is mapped to an embedding: an m-dimensional feature vector;
  2. a probability function over word sequences is expressed in terms of these vectors;

[Bengio et al., 2003]

Feed-forward NLM

- With NN we can exploit distributed representations to allow for statistical weight sharing.
- How does it work:
  1. each word is mapped to an embedding: an m-dimensional feature vector;
  2. a probability function over word sequences is expressed in terms of these vectors;
  3. we jointly learn the feature vectors and the parameters of the probability function.

[Bengio et al., 2003]

Feed-forward NLM

- Similar words are expected to have similar feature vectors: (dog,cat), (running,walking), (bedroom,room)

Feed-forward NLM

- Similar words are expected to have similar feature vectors: (dog,cat), (running,walking), (bedroom,room)
- With this, probability mass is naturally transferred from (1) to (2):

Feed-forward NLM

- Similar words are expected to have similar feature vectors: (dog,cat), (running,walking), (bedroom,room)
- With this, probability mass is naturally transferred from (1) to (2):
- The cat is walking in the bedroom.

Feed-forward NLM

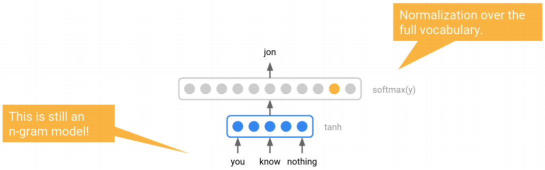
- Similar words are expected to have similar feature vectors: (dog,cat), (running,walking), (bedroom,room)
- With this, probability mass is naturally transferred from (1) to (2):
- The cat is walking in the bedroom.
- The dog is running in the room.

Feed-forward NLM

- Similar words are expected to have similar feature vectors: (dog,cat), (running,walking), (bedroom,room)
- With this, probability mass is naturally transferred from (1) to (2):
- The cat is walking in the bedroom.
- The dog is running in the room.
- Take-away message:
  The presence of only one sentence in the training data will increase the probability of a combinatorial number of neighbours in sentence space.

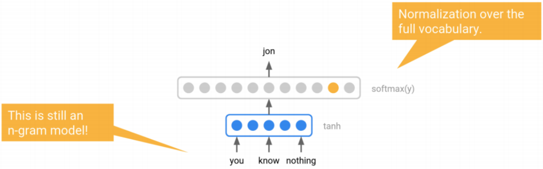
Feed-forward NLM

- FF-LM



Feed-forward NLM

- FF-LM

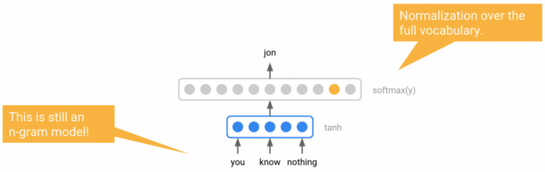


- $\boldsymbol{E}_{\text{you}}, \boldsymbol{E}_{\text{know}}, \boldsymbol{E}_{\text{nothing}} \in \mathbb{R}^{100}$

$$\begin{aligned} \boldsymbol{x} &= [\boldsymbol{E}_{\text{you}}; \boldsymbol{E}_{\text{know}}; \boldsymbol{E}_{\text{nothing}}] \in \mathbb{R}^{300} \\ \boldsymbol{y} &= \boldsymbol{W}_3 \tanh(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{W}_2\boldsymbol{x} + \boldsymbol{b}_2 \end{aligned} \tag{8}$$

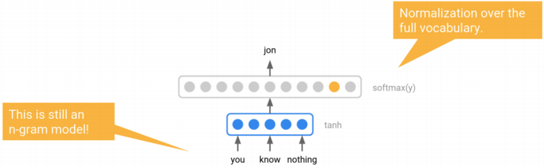
Feed-forward NLM

- FF-LM



Feed-forward NLM

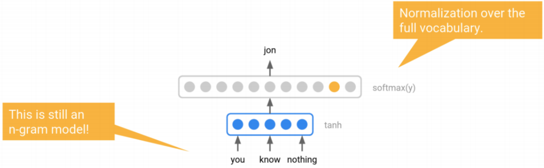
- FF-LM



- The non-linear activation functions perform feature combinations that a linear model cannot do;

Feed-forward NLM

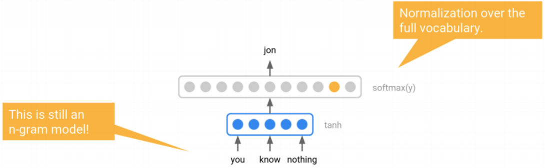
- FF-LM



- The non-linear activation functions perform feature combinations that a linear model cannot do;
- End-to-end training on next word prediction.

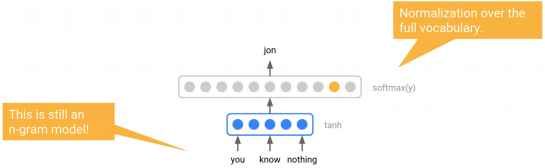
Feed-forward NLM

- FF-LM



Feed-forward NLM

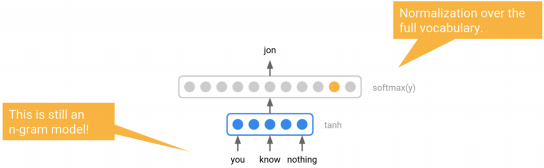
- FF-LM



- We now have much better generalisation, but still a limited history/context.

Feed-forward NLM

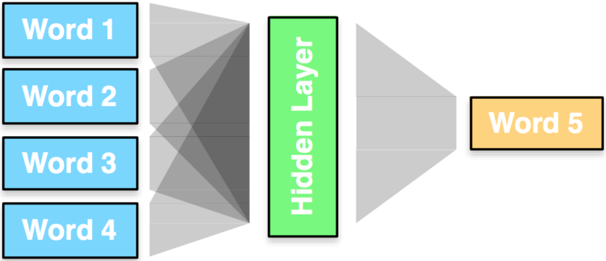
- FF-LM



- We now have much better generalisation, but still a limited history/context.
- Recurrent neural networks have unlimited history

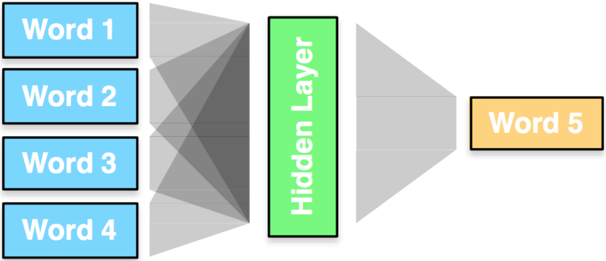
Feed-forward NLM

- FF-LM



Feed-forward NLM

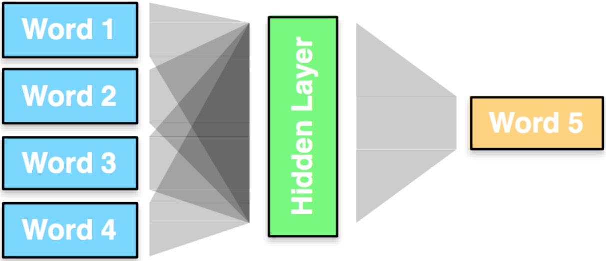
- FF-LM



- We now have much better generalisation, but still a limited history/context.

Feed-forward NLM

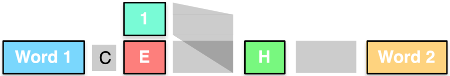
- FF-LM



- We now have much better generalisation, but still a limited history/context.
- Recurrent neural networks have unlimited history

RNN NLM

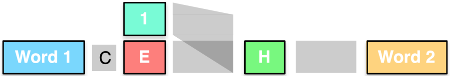
- RNN-LM



[Mikolov et al., 2010]

RNN NLM

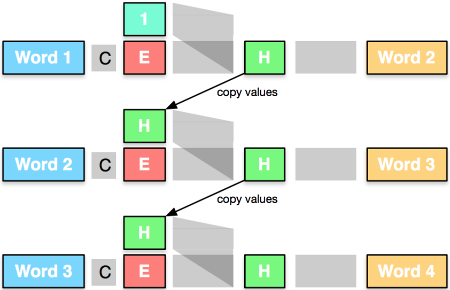
- RNN-LM



- Start: predict second word from first

[Mikolov et al., 2010]

RNN NLM


1
Word 1
C
E
H
Word 2
copy values
H
Word 2
C
E
H
Word 3
copy values
H
Word 3
C
E
H
Word 4

Outline

Sen VAE

- Model observations as draws from the marginal of a DGM.

[Bowman et al., 2015]

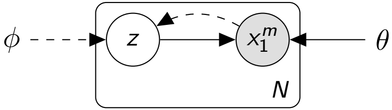
Sen VAE

- Model observations as draws from the marginal of a DGM.
- An NN maps from a latent sentence embedding $z \in R^{dz}$ to a distribution $p(x|z, \theta)$ over sentences,

$$\begin{aligned} p(x|\theta) &= \int p(z)p(x|z,\theta)\mathrm{d}z \\ &= \int \mathcal{N}(z|0,I)\prod_{i=1}^{|x|} \mathrm{Cat}\left(x_i|f\left(z,x_{<i};\theta\right)\right)\mathrm{d}z \end{aligned} \tag{9}$$

[Bowman et al., 2015]

Sen VAE



Generative model

- $Z \sim \mathcal{N}(0, I)$
- $X_i | z, x_{<i} \sim \mathrm{Cat}(f_\theta(z, x_{<i}))$

Inference model

- $Z \sim \mathcal{N}(\mu_\phi(x_1^m), \sigma_\phi(x_1^m)^2)$

Sen VAE

- Generation is one word at a time without Markov assumptions, but $f()$conditions on z
  in addition to the observed prefix.

Sen VAE

- Generation is one word at a time without Markov assumptions, but $f()$conditions on z in addition to the observed prefix.
- The conditional $p(x|z,\theta)$ is the decoder.

Sen VAE

- Generation is one word at a time without Markov assumptions, but $f()$conditions on z in addition to the observed prefix.
- The conditional $p(x|z,\theta)$ is the decoder.
- $p(x|\theta)$ is the marginal likelihood.

Sen VAE

- Generation is one word at a time without Markov assumptions, but $f()$conditions on z in addition to the observed prefix.
- The conditional $p(x|z,\theta)$ is the decoder.
- $p(x|\theta)$ is the marginal likelihood.
- We train the model to assign high (marginal) probability to observations like a LMs.

Sen VAE

- Generation is one word at a time without Markov assumptions, but $f()$conditions on z in addition to the observed prefix.
- The conditional $p(x|z,\theta)$ is the decoder.
- $p(x|\theta)$ is the marginal likelihood.
- We train the model to assign high (marginal) probability to observations like a LMs.
- However the model uses a latent space to exploit neighbourhood and smoothness in latent space to capture regularities in data space.
  For example, it may group sentences according to certain e.g. lexical choices, syntactic complexity, lexical semantics, etc...

Approximate Inference

- The model has a diagonal Gaussian distribution as variational posterior:
$q_\phi(z|x) = \mathcal{N}\left(z|\mu_\phi(x), \mathrm{diag}\left(\sigma_\phi(x)\right)\right)$

Approximate Inference

- The model has a diagonal Gaussian distribution as variational posterior:
  $q_\phi(z|x) = \mathcal{N}\left(z|\mu_\phi(x), \operatorname{diag}\left(\sigma_\phi(x)\right)\right)$
- With reparametrisation:
  $z = h_\phi(\epsilon, x) = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \qquad \text{where } \epsilon \sim \mathcal{N}\left(0, \mathbf{I}\right) f$

Approximate Inference

- The model has a diagonal Gaussian distribution as variational posterior:
  $q_\phi(z|x) = \mathcal{N}\left(z|\mu_\phi(x), \mathrm{diag}\left(\sigma_\phi(x)\right)\right)$
- With reparametrisation:
  $z = h_\phi(\epsilon, x) = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \qquad \text{where } \epsilon \sim \mathcal{N}\left(0, \mathbf{I}\right) f$
- Analytical KL:
  $\mathrm{KL}\left[q_\phi(z|x) \| p_\theta(z)\right] =$
  $\frac{1}{2}\sum_{d=1}^{\mathrm{D}_z}\left(-\log\sigma_\phi^2(x) - 1 + \sigma_\phi^2(x) + \mu_\phi^2(x)\right)$

Approximate Inference

- We jointly estimate the parameters of both generative and inference by maximising a lowerbound on the log-likelihood function (ELBO):

$$\mathcal{L}(\theta, \phi|x) = \mathbb{E}_{q(z|x,\phi)}[\log p(x|z,\theta)]. \\ - \mathrm{KL}(q(z|x,\phi)|p(z)) \tag{10}$$

Architecture

- Gaussian Sen VAE parametrises a categorical distribution over the vocabulary for each given prefix, and, it conditions on a latent embedding:
  $Z \sim \mathcal{N}(0, I)$,
  $X_i | z, x_{<i} \sim \mathrm{Cat}\left(f\left(z, x_{<i}; \theta\right)\right)$

Architecture

- Gaussian Sen VAE parametrises a categorical distribution over the vocabulary for each given prefix, and, it conditions on a latent embedding:
  $Z \sim \mathcal{N}(0, I)$,
  $X_i | z, x_{<i} \sim \mathrm{Cat}\left(f\left(z, x_{<i}; \theta\right)\right)$
-
$$\begin{aligned} f\left(z, x_{<i}; \theta\right) &= \mathrm{softmax}\left(\mathbf{s}_i\right) \\ \mathbf{e}_i &= \mathrm{emb}\left(x_i; \theta_{\mathrm{emb}}\right) \\ \mathbf{h}_0 &= \tanh\left(\mathrm{affine}\left(z; \theta_{\text{ init }}\right)\right) \\ \mathbf{h}_i &= \mathrm{GRU}\left(\mathbf{h}_{i-1}, \mathbf{e}_{i-1}; \theta_{\mathrm{gru}}\right) \\ \mathbf{s}_i &= \mathrm{affine}\left(\mathbf{h}_i; \theta_{\mathrm{out}}\right) \end{aligned} \tag{11}$$

The Strong Decoder Problem

- The VAE may ignore the latent variable given the interaction between the prior and posterior in the KL divergence.

The Strong Decoder Problem

- The VAE may ignore the latent variable given the interaction between the prior and posterior in the KL divergence.
- This problem appears when we have strong decoders conditional likelihoods $p(x|z)$ parametrised by high capacity models

The Strong Decoder Problem

- The VAE may ignore the latent variable given the interaction between the prior and posterior in the KL divergence.
- This problem appears when we have strong decoders conditional likelihoods $p(x|z)$ parametrised by high capacity models
- The model might achieve a high ELBO without using information from z

The Strong Decoder Problem

- The VAE may ignore the latent variable given the interaction between the prior and posterior in the KL divergence.
- This problem appears when we have strong decoders conditional likelihoods $p(x|z)$ parametrised by high capacity models
- The model might achieve a high ELBO without using information from z
- RNN LM is strong decoder because they condition on all previous context when generating a word

What to do?

- Weakening the Decoder, the model relies on the latent variables the reconstruction of the observed data.

What to do?

- Weakening the Decoder, the model relies on the latent variables the reconstruction of the observed data.
- KL Annealing, weigh the KL term in the ELBO with a factor that is annealed from 0 to 1 over a fixed number of steps of size $\gamma \in (0, 1)$

What to do?

- Weakening the Decoder, the model relies on the latent variables the reconstruction of the observed data.
- KL Annealing, weigh the KL term in the ELBO with a factor that is annealed from 0 to 1 over a fixed number of steps of size $\gamma \in (0, 1)$
- Word Dropout, by dropping a percentage of the input at random, the decoder has to rely on the latent variable to fill in the missing gaps.

What to do?

- Weakening the Decoder, the model relies on the latent variables the reconstruction of the observed data.
- KL Annealing, weigh the KL term in the ELBO with a factor that is annealed from 0 to 1 over a fixed number of steps of size $\gamma \in (0, 1)$
- Word Dropout, by dropping a percentage of the input at random, the decoder has to rely on the latent variable to fill in the missing gaps.
- Freebits because it allows encoding the first r nats of information for free.
  $\max(r, \mathrm{KL}(q_\phi(z|x) \| p(z)))$

Metrics

- For VAE models, Negative log-likelihood NLL is estimated, since we do not have access to the exact marginal likelihood.

Metrics

- For VAE models, Negative log-likelihood NLL is estimated, since we do not have access to the exact marginal likelihood.
- We use an importance sampling (IS) estimate://

$$
\begin{aligned}
p(x|\theta) = \int p(z, x|\theta)\mathrm{d}z \stackrel{\mathrm{IS}}{=} \int q(z|x)\frac{p(z,x|\theta)}{q(z|x)}\mathrm{d}z \\
\approx \frac{1}{S}\sum_{s=1}^{S} \frac{p\left(z^{(s)},x|\theta\right)}{q\left(z^{(s)}|x\right)} \text{ where } z^{(s)} \sim q(z|x)
\end{aligned}
\tag{12}
$$

Metrics

- For VAE models, Negative log-likelihood NLL is estimated, since we do not have access to the exact marginal likelihood.
- We use an importance sampling (IS) estimate://

$$p(x|\theta) = \int p(z, x|\theta)\mathrm{d}z \overset{\text{IS}}{=} \int q(z|x)\frac{p(z,x|\theta)}{q(z|x)}\mathrm{d}z \approx \frac{1}{S}\sum_{s=1}^{S} \frac{p\left(z^{(s)},x|\theta\right)}{q\left(z^{(s)}|x\right)} \text{ where } z^{(s)} \sim q(z|x) \tag{12}$$

- Perplexity PPL: the exponent of average per-word entropy, given N i.i.d. sequences

$$\mathrm{PPL} = \exp\left(\frac{\sum_{i=1}^{N} \log p\left(x_i\right)}{\sum_{i=1}^{N} |x_i|}\right) \tag{13}$$

Metrics

- For VAE models, Negative log-likelihood NLL is estimated, since we do not have access to the exact marginal likelihood.
- We use an importance sampling (IS) estimate://

$$p(x|\theta) = \int p(z,x|\theta)\mathrm{d}z \stackrel{\text{IS}}{=} \int q(z|x)\frac{p(z,x|\theta)}{q(z|x)}\mathrm{d}z \\ \approx \frac{1}{S}\sum_{s=1}^{S} \frac{p\left(z^{(s)},x|\theta\right)}{q\left(z^{(s)}|x\right)} \text{ where } z^{(s)} \sim q(z|x) \tag{12}$$

- Perplexity PPL: the exponent of average per-word entropy, given N i.i.d. sequences

$$\mathrm{PPL} = \exp\left(\frac{\sum_{i=1}^{N} \log p\left(x_i\right)}{\sum_{i=1}^{N} |x_i|}\right) \tag{13}$$

- perplexity is based on the importance sampled NLL

Baseline

- RNNLM (Dyer et al., 2016)

Baseline

- RNNLM (Dyer et al., 2016)
- At each step, an RNNLM parameterises a categorical distribution over the vocabulary, i.e. $X_i | x_{<i} \sim \mathrm{Cat}\left(f\left(x_{<i}; \theta\right)\right)$ and

$$\begin{aligned} f\left(x_{<i}; \theta\right) &= \mathrm{softmax}\left(\mathbf{s}_i\right) \text{ and} \\ \mathbf{e}_i &= \mathrm{emb}\left(x_i; \theta_{\text{emb}}\right) \\ \mathbf{h}_i &= \mathrm{GRU}\left(\mathbf{h}_{i-1}, \mathbf{e}_{i-1}; \theta_{\text{gru}}\right) \\ \mathbf{s}_i &= \mathrm{affine}\left(\mathbf{h}_i; \theta_{\text{out}}\right) \end{aligned} \tag{14}$$

Baseline

- RNNLM (Dyer et al., 2016)
- At each step, an RNNLM parameterises a categorical distribution over the vocabulary, i.e. $X_i|x_{<i} \sim \mathrm{Cat}\left(f\left(x_{<i};\theta\right)\right)$ and

$$\begin{aligned} f\left(x_{<i};\theta\right) &= \mathrm{softmax}\left(\mathbf{s}_i\right) \text{ and} \\ \mathbf{e}_i &= \mathrm{emb}\left(x_i;\theta_{\mathrm{emb}}\right) \\ \mathbf{h}_i &= \mathrm{GRU}\left(\mathbf{h}_{i-1},\mathbf{e}_{i-1};\theta_{\mathrm{gru}}\right) \\ \mathbf{s}_i &= \mathrm{affine}\left(\mathbf{h}_i;\theta_{\mathrm{out}}\right) \end{aligned} \tag{14}$$

- Embedding layer (emb), one (or more) GRU cell(s) ($h_0 \in \theta$ is a parameter of the model), and an affine layer to map from the dimensionality of the GRU to the vocabulary size.

Data

- Wall Street Journal part of the Penn Treebank corpus

Data

- Wall Street Journal part of the Penn Treebank corpus
- Preprocessing on train-validation-test split as [Dyer et al., 2016]

Data

- Wall Street Journal part of the Penn Treebank corpus
- Preprocessing on train-validation-test split as [Dyer et al., 2016]
- WSJ section 1-21 training,

Data

- Wall Street Journal part of the Penn Treebank corpus
- Preprocessing on train-validation-test split as [Dyer et al., 2016]
- WSJ section 1-21 training,
- Section 23 as test corpus

Data

- Wall Street Journal part of the Penn Treebank corpus
- Preprocessing on train-validation-test split as [Dyer et al., 2016]
- WSJ section 1-21 training,
- Section 23 as test corpus
- Section 24 as validation

Results

	NLL↓	PPL↓
RNN-LM	118.7±0.12	107.1±0.46
VAE	118.4±0.09	105.7±0.36
Annealing	117.9±0.08	103.7±0.31
Free-bits	117.5±0.18	101.9±0.77

Samples

- decode greedily from a prior sampl and the variability is due to the generator's reliance on the latent sample.

Samples

- decode greedily from a prior sampl and the variability is due to the generator's reliance on the latent sample.
- The VAE ignores z and greedy generation from a prior sample is essentially deterministic in that case

Sample	Closest training instance
For example, the Dow Jones Industrial Average fell almost 80 points to close at 2643.65.	*By futures-related program buying, the Dow Jones Industrial Average gained 4.92 points to close at 2643.65.*
The department store concern said it expects to report profit from continuing operations in 1990.	*Rolls-Royce Motor Cars Inc. said it expects its U.S. sales to remain steady at about 1,200 cars in 1990.*
The new U.S. auto makers say the accord would require banks to focus on their core businesses of their own account.	*International Minerals said the sale will allow Mallinckrodt to focus its resources on its core businesses of medical products, specialty chemicals and flavors.*

Samples

- Homotopy, decode greedily from points lying between a posterior sample conditioned on the first sentence and a posterior sample conditioned on the last sentence.

Samples

- Homotopy, decode greedily from points lying between a posterior sample conditioned on the first sentence and a posterior sample conditioned on the last sentence.
- $z_\alpha = \alpha * z_1 + (1 - \alpha) * z_2$
  $\alpha \in [0, 1]$

The inquiry soon focused on the judge.
The judge declined to comment on the floor.
The judge was dismissed as part of the settlement.
The judge was sentenced to death in prison.
The announcement was filed against the SEC.
The offer was misstated in late September.
The offer was filed against bankruptcy court in New York.
The letter was dated Oct. 6.

References I

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003. URL http://dblp.uni-trier.de/db/journals/jmlr/jmlr3.html#BengioDVJ03.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. Generating sentences from a continuous space. *CoRR*, abs/1511.06349, 2015. URL http://arxiv.org/abs/1511.06349.

References II

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1024. URL https://www.aclweb.org/anthology/N16-1024.

Joshua T. Goodman. A bit of progress in language modeling. *Comput. Speech Lang.*, 15(4):403–434, October 2001. ISSN 0885-2308. doi: 10.1006/csla.2001.0174. URL http://dx.doi.org/10.1006/csla.2001.0174.

References III

Fred Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In Edzard S. Gelsema and Laveen N. Kanal, editors, *Proceedings, Workshop on Pattern Recognition in Practice*, pages 381–397. North Holland, Amsterdam, 1980.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In Takao Kobayashi, Keikichi Hirose, and Satoshi Nakamura, editors, *INTERSPEECH*, pages 1045–1048. ISCA, 2010. URL `http://dblp.uni-trier.de/db/conf/interspeech/interspeech2010.html#MikolovKBCK10`.