ETH zürich

Exploring the Limits of Language Modeling

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, Yonghui Wu
Google Brain

Presented by Gabriel Gavrilas

Ernest Shackleton on his first Antarctica Expedition

Agenda

1. Context & Motivation
2. Background
3. Method
4. Experiments & Results
5. Commentary & Discussion

Context

- Language Modeling (2016)
  - Task specific
  - Small datasets
  - Small models
  - N-grams

Context

- Language Modeling (2016)
  - Task specific
  - Small datasets
  - Small models
  - N-grams

- This work:
  - Language modeling as an independent task
  - Large Datasets
  - Large Models
  - Superiority of RNNs

Motivation

- Explore recent advances in RNNs for large scale Language Modeling
- Focus on two key challenges:
  - Corpora and vocabulary sizes
  - Complex, long term structure of language

Background

Background - Language Modeling

„The goal of LM is to learn a probability distribution over sequences of symbols pertaining to a language“

- RNN-based LMs employ the chain rule to model joint probabilities over word sequences:

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

- In this paper they use LSTM-based LMs for all experiments

Background - Vocabulary Size

- Vocabulary of a dataset: The set of unique words
- Vocabulary of a model: The set of supported words[1]

[1] https://machinetranslate.org/vocabulary

Background - Vocabulary Size

- Vocabulary of a dataset: The set of unique words
- Vocabulary of a model: The set of supported words[1]
- One Billion Word Benchmark (Chelba et al., 2014)
  - 0.8B Words
  - Vocabulary of 793’471 words

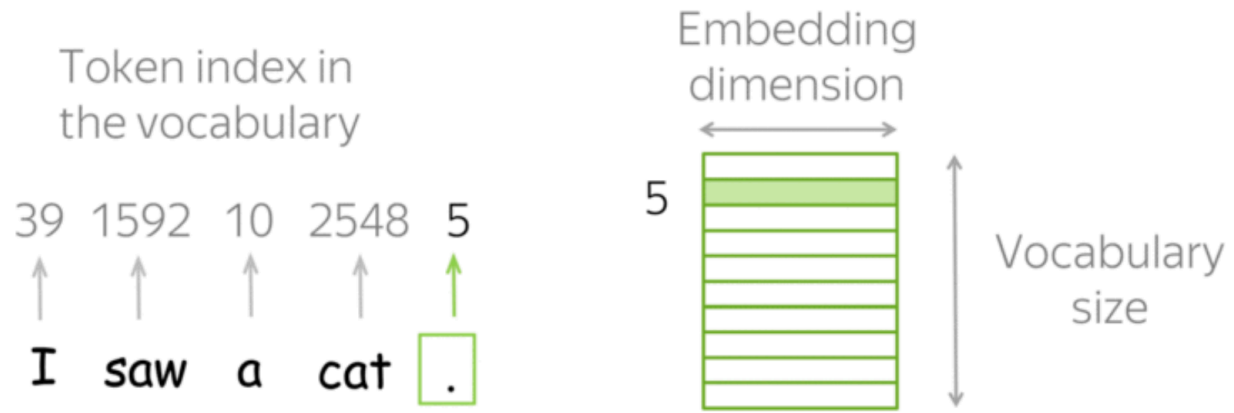
[1] https://machinetranslate.org/vocabulary

Background - Vocabulary Size

- Vocabulary of a dataset: The set of unique words
- Vocabulary of a model: The set of supported words[1]
- One Billion Word Benchmark (Chelba et al., 2014)
  - 0.8B Words
  - Vocabulary of 793’471 words
- 3 Places where the vocabulary size matters in RNN LMs:
  - Input embeddigs
  - Model output
  - Softmax

[1] https://machinetranslate.org/vocabulary

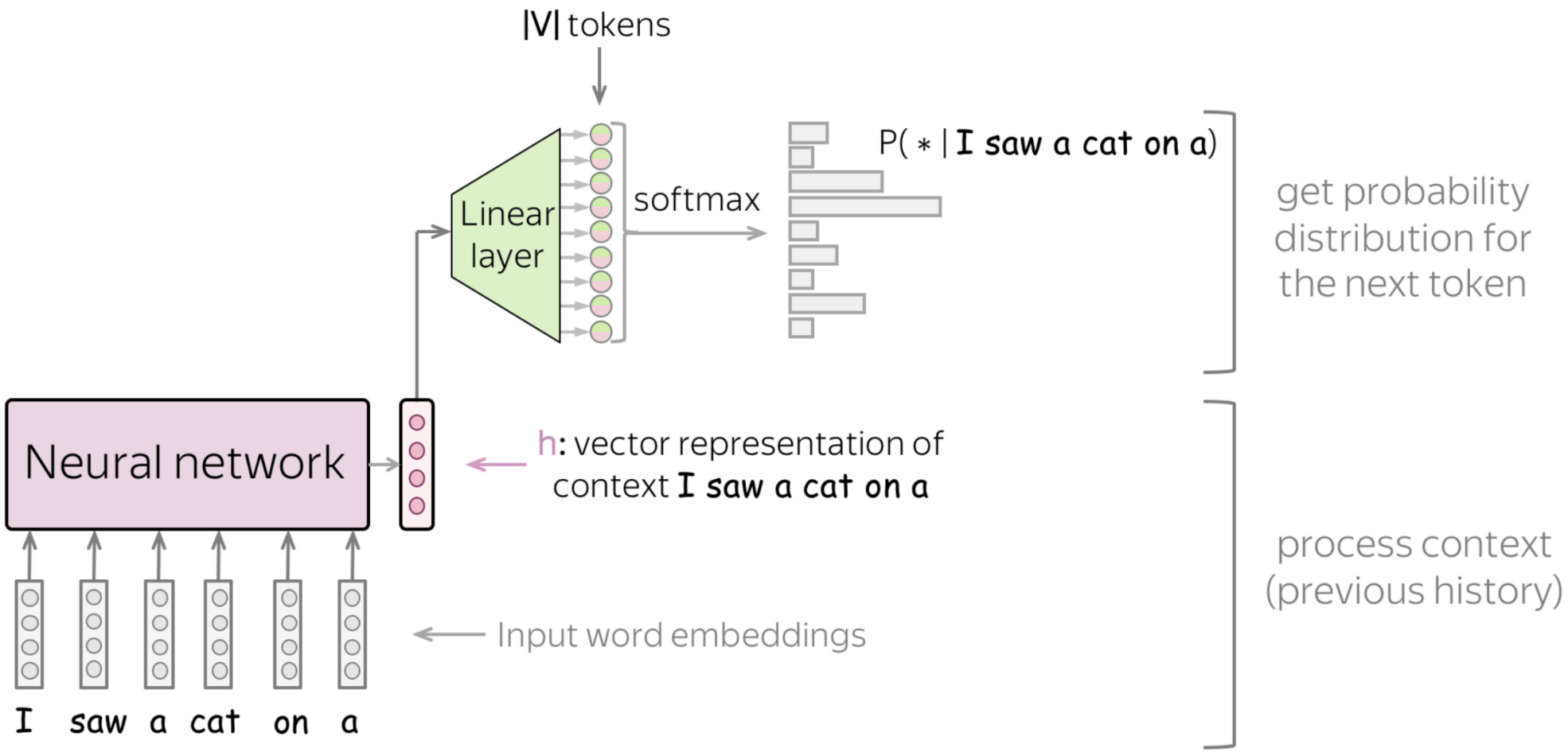
Background - Vocabulary Size



- Input embeddings
  - Computed using an embedding layer of size $|V| \times |h|$
  - For large $|V|$ the embedding weights can make up a majority of the model parameters

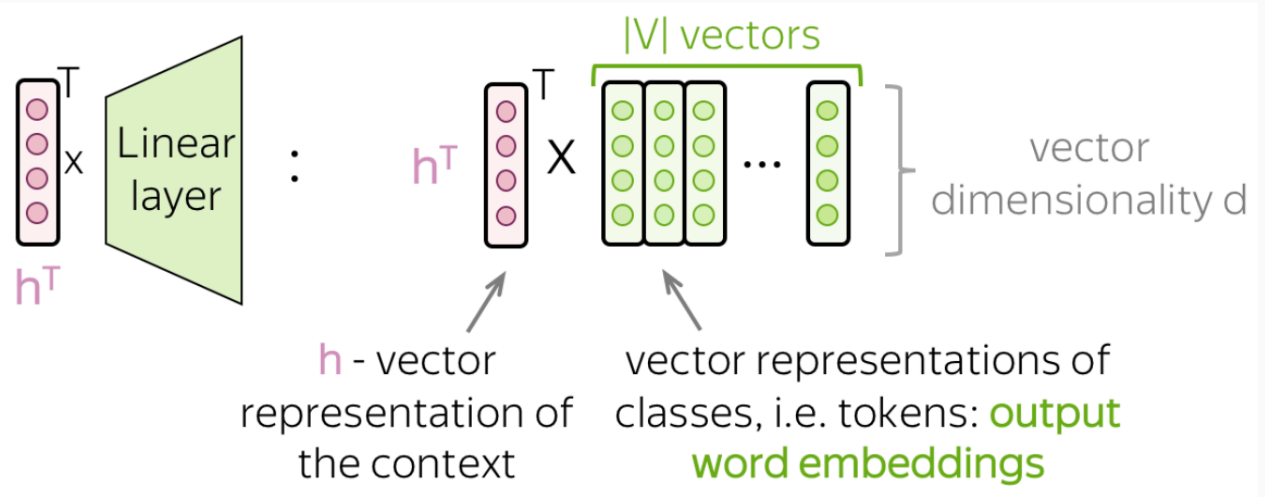
https://lena-voita.github.io/nlp_course/language_modeling.html

Background - Vocabulary Size



https://lena-voita.github.io/nlp_course/language_modeling.html

Background - Vocabulary Size



- Model output
  - Computing the inner product for all embeddings is expensive
  - „Output embeddings“ again make up a large part of the model for large $|V|$

Background - Vocabulary Size

- Softmax:

$$p(w) = \frac{\exp(z_w)}{\sum_{w' \in V} \exp(z_{w'})} \quad , \quad z_w = h^T e_w$$

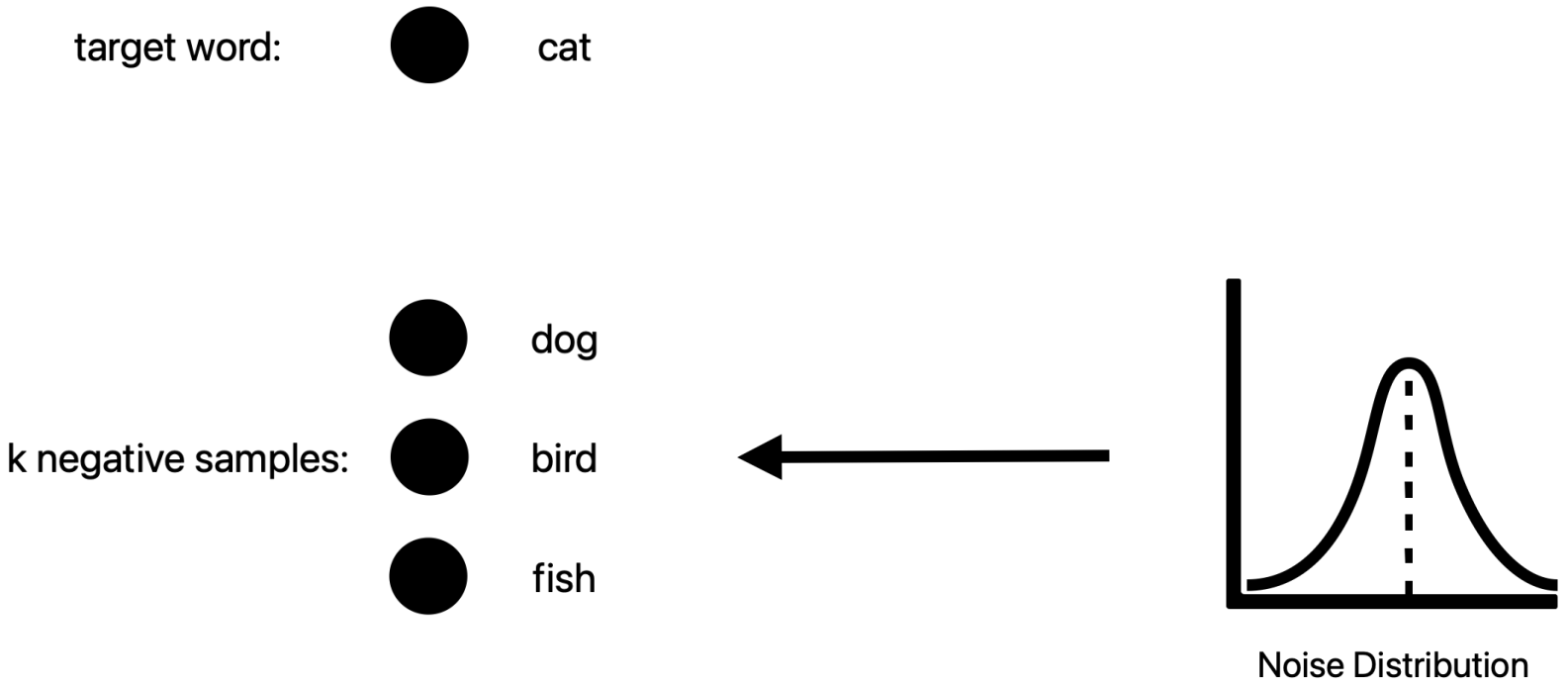
  - Context vector h
  - „Output embedding“ e_w for word w
- For large $|V|$ computing the sum in the denominator (partition function) becomes prohibitive

Method

Method - Noise Contrastive Estimation & Importance Sampling

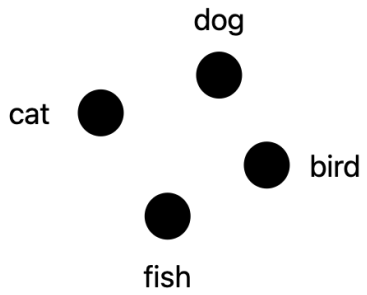
- Solutions to the Softmax scaling issue:
  - Importance Sampling
  - Noise Contrastive Estimation
  - Hierarchical Softmax
  - Self normalizing partition functions

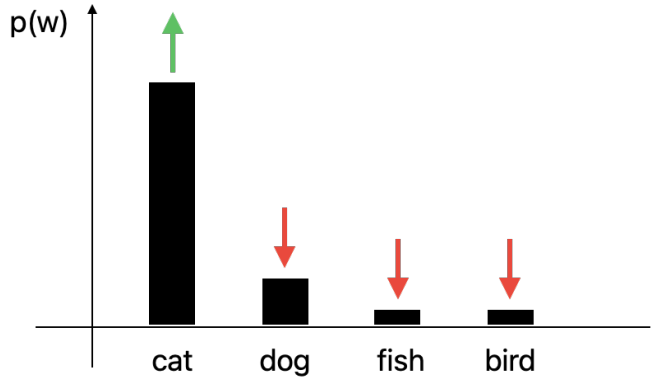
Method - Noise Contrastive Estimation & Importance Sampling



Method - Noise Contrastive Estimation & Importance Sampling

Negative Sampling





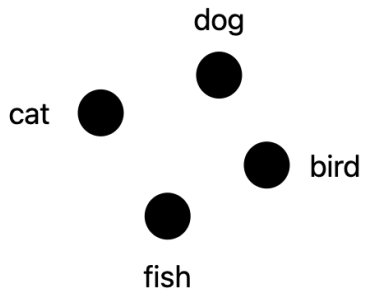
Method - Noise Contrastive Estimation & Importance Sampling

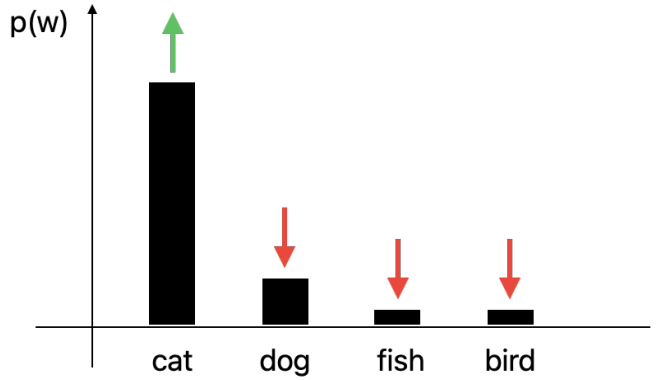
- Negative Sampling
  - Sample k words form a noise distribution over the vocabulary
  - Approximate the partition function by summing only over the samples and the target word
  - Calculate the Softmax only for the k noise samples and the target word

https://www.kdnuggets.com/2019/07/introduction-noise-contrastive-estimation.html

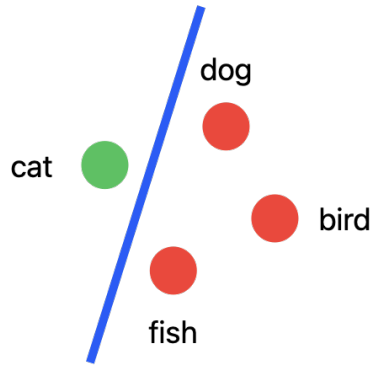
Method - Noise Contrastive Estimation & Importance Sampling

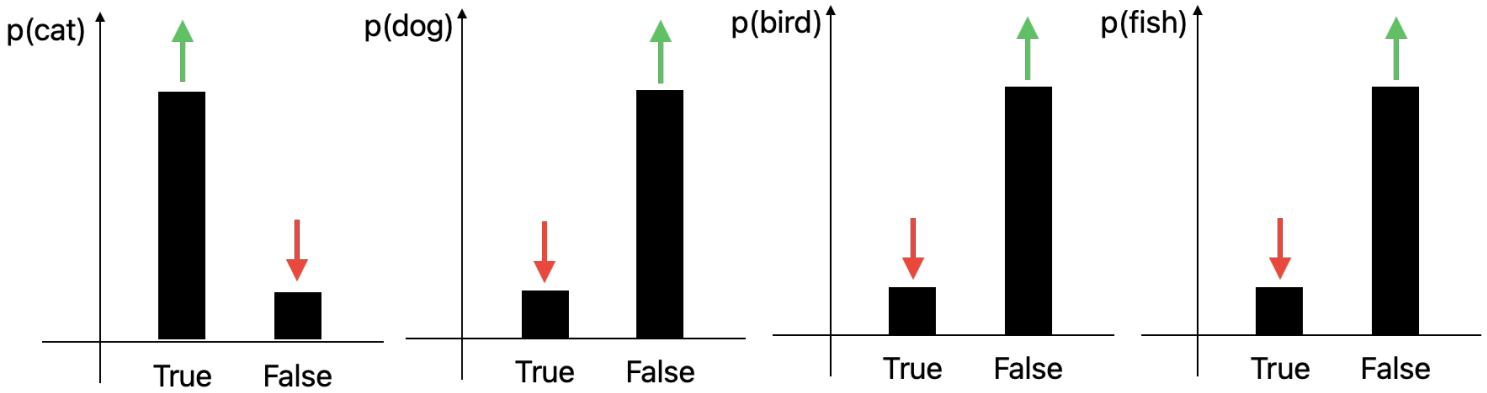
Negative Sampling





Noise Contrastive Estimation





Method - Noise Contrastive Estimation & Importance Sampling

- Noise Contrastive Estimation:
  - Binary task: Discriminate between samples from the true distribution p_d and a noise distribution p_n.

$$p(Y = true|w) = \frac{p_d(w)}{p_d(w) + kp_n(w)}$$

Method - Noise Contrastive Estimation & Importance Sampling

- Noise Contrastive Estimation:
  - Binary task: Discriminate between samples from the true distribution p_d and a noise distribution p_n.

$$p(Y = true|w) = \frac{p_d(w)}{p_d(w) + kp_n(w)}$$

  - Approximate $\log p_d(w)$ with an RNN $s_\theta(w, h)$ and train a logistic classifier on the binary objective:

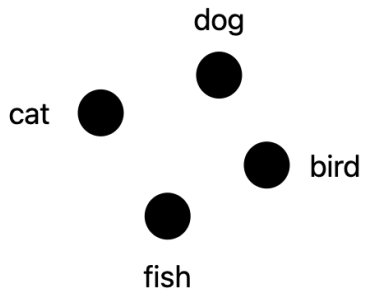
$$p_\theta(Y = true|w) = \sigma(s_\theta(w, h) - \log kp_n(w))$$

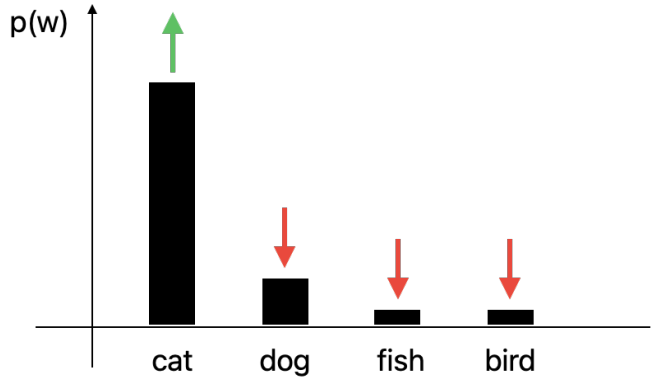
  - Good approximation of $p_d(w)$:

$$p'(w) = softmax(s_\theta(w, h))$$

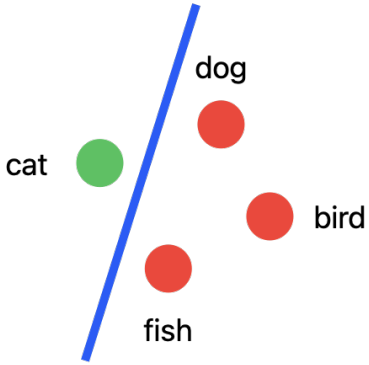
Method - Noise Contrastive Estimation & Importance Sampling

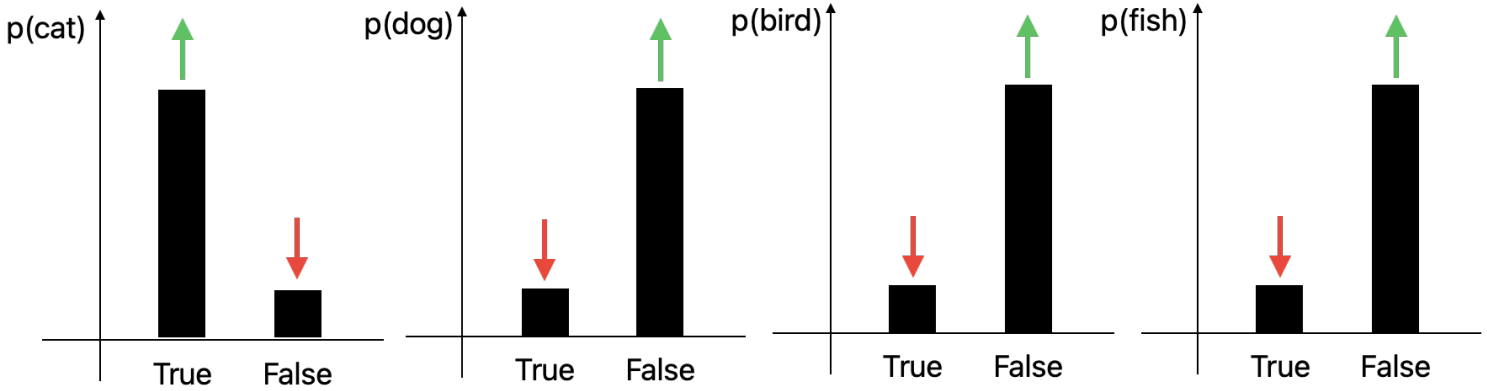
Negative Sampling



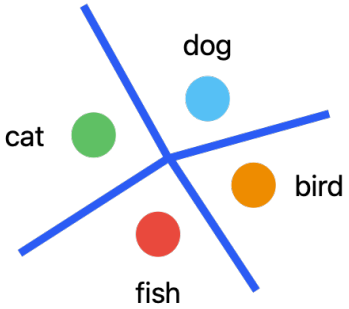


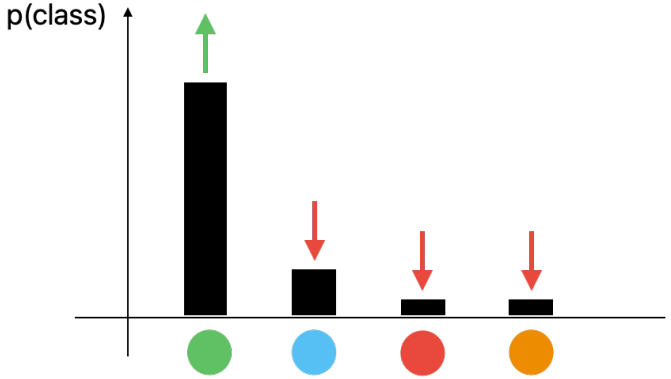
Noise Contrastive Estimation





Importance Sampling





Method - Noise Contrastive Estimation & Importance Sampling

- Importance Sampling
  - Multi-class setting: Define a set $W = \{w_1, \ldots, w_{k+1}\}$ with k noise samples and the true data sample.
  - Bayes rule: $p(Y = k|W) \propto_Y \frac{p_d(w_k)}{p_n(w_k)}$
  - Multi-class loss: $p(Y = k|W) = softmax(s_\theta(w_k) - \log p_n(w_k))$
  - Maximizing $\log p(Y = 1|W)$ leads to a good approximation of $p_d(w)$:

$$p'(w) = softmax(s_\theta(w, h))$$

Method - Noise Contrastive Estimation & Importance Sampling

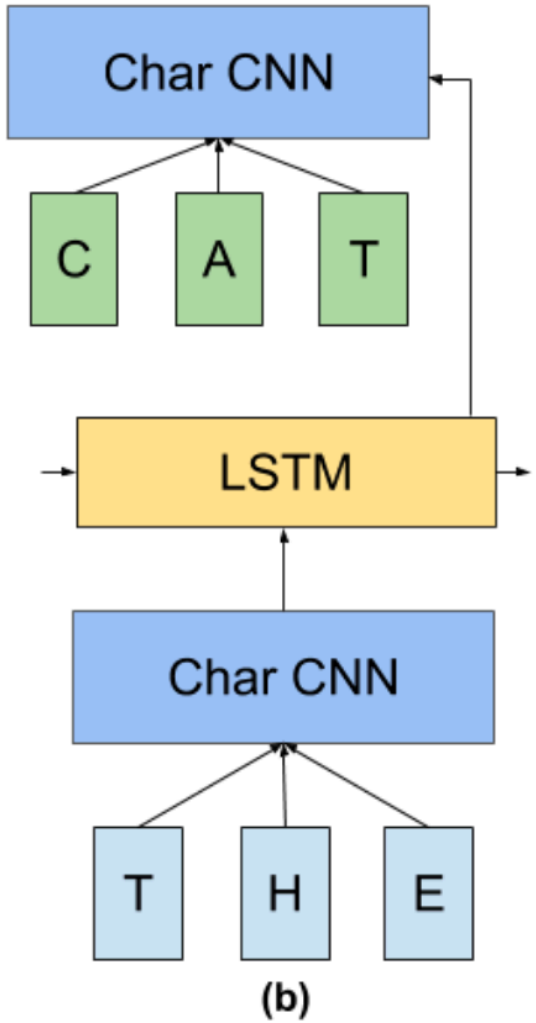
- Methods are related
- Advantages both methods:
  - Cheap
  - Less model parameter updates
- Updates to logits are tied with IS
- Can only be used during training

Method - CNN Softmax

- Kim et al., 2015 used character-level CNNs for input embeddings
  - Incorporate morphological knowledge
  - Better performance
  - Reduces size of model drastically
- They propose using character-level embeddings also for the output embeddings:

$$z_w = h^T e_w, \quad e_w = CNN(chars_w)$$

Method - CNN Softmax



- Details:
  - Input and output CNNs trained separately
  - Pre-compute embeddings for inference
  - Improve performance using low-dimensional correction embedding learnt per word:

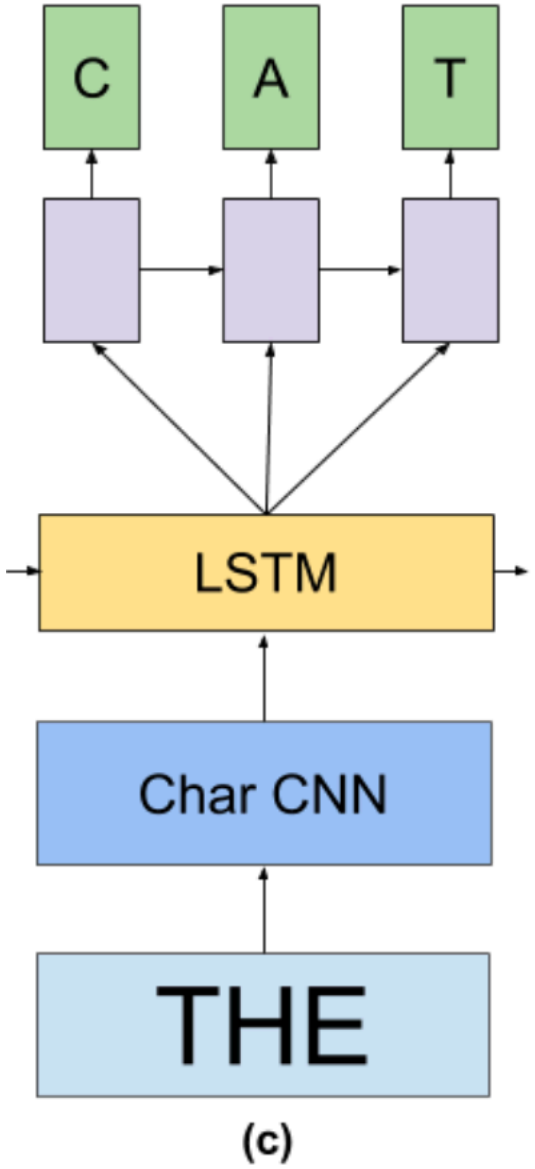
$$z_w = h^T CNN(chars_w) + h^T M corr_w$$

Method - CNN Softmax

- Advantages:
  - Reduces amount of parameters
  - Incorporate morphological knowledge
  - Useful in combination with sampling
  - Able to handle OOV words
  - Trade-off accuracy and model size by correction term

Method - Char LSTM Predictions

- CNN Softmax slow during inference
- Character-level LSTMs
  - small vocabulary
  - hard to train
- Hierarchical LSTM:
  - Feed word-level LSTM hidden state into character-level LSTM



(c)

Experiments & Results

Training

- Setup:
  - TensorFlow
  - Truncated BPTT for 20 steps and gradient clipping
  - Importance sampling with 8192 negative samples per step and batch
  - 32 GPUs with asynchronous gradient updates
- Best model trained for 3 week and beat SOTA after 2h
- Report perplexity and model size

Results

Model	Test Perplexity	Number of Params [billions]
Sigmoid-RNN-2048 (Ji et al., 2015a)	68.3	4.1
Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)	67.6	1.76
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33
RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (No Dropout)	37.9	3.3
LSTM-8192-2048 (50% Dropout)	32.2	3.3
2-Layer LSTM-8192-1024 (BIG LSTM)	30.6	1.8
BIG LSTM+CNN Inputs	**30.0**	**1.04**
BIG LSTM+CNN Inputs + CNN Softmax	39.8	**0.29**
BIG LSTM+CNN Inputs + CNN Softmax + 128-dim correction	35.8	**0.39**
BIG LSTM+CNN Inputs + Char LSTM predictions	47.9	**0.23**

- Baselines on 1B Word Benchmark

Results

Model	Test Perplexity	Number of Params [billions]
Sigmoid-RNN-2048 (Ji et al., 2015a)	68.3	4.1
Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)	67.6	1.76
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33
RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (No Dropout)	37.9	3.3
LSTM-8192-2048 (50% Dropout)	32.2	3.3
2-Layer LSTM-8192-1024 (BIG LSTM)	30.6	1.8
BIG LSTM+CNN Inputs	**30.0**	**1.04**
BIG LSTM+CNN Inputs + CNN Softmax	39.8	**0.29**
BIG LSTM+CNN Inputs + CNN Softmax + 128-dim correction	35.8	**0.39**
BIG LSTM+CNN Inputs + Char LSTM predictions	47.9	**0.23**

- Larger models achieve lower perplexity
- Dropout increases performance

Results

Model	Test Perplexity	Number of Params [billions]
Sigmoid-RNN-2048 (Ji et al., 2015a)	68.3	4.1
Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)	67.6	1.76
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33
RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (No Dropout)	37.9	3.3
LSTM-8192-2048 (50% Dropout)	32.2	3.3
2-Layer LSTM-8192-1024 (BIG LSTM)	30.6	1.8
BIG LSTM+CNN Inputs	**30.0**	**1.04**
BIG LSTM+CNN Inputs + CNN Softmax	39.8	**0.29**
BIG LSTM+CNN Inputs + CNN Softmax + 128-dim correction	35.8	**0.39**
BIG LSTM+CNN Inputs + Char LSTM predictions	47.9	**0.23**

- Best single model: 2-layer LSTM with CNN inputs
- Less parameters due to CNN inputs

Results

Model	Test Perplexity	Number of Params [billions]
Sigmoid-RNN-2048 (Ji et al., 2015a)	68.3	4.1
Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)	67.6	1.76
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33
RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (No Dropout)	37.9	3.3
LSTM-8192-2048 (50% Dropout)	32.2	3.3
2-Layer LSTM-8192-1024 (BIG LSTM)	30.6	1.8
BIG LSTM+CNN Inputs	**30.0**	**1.04**
BIG LSTM+CNN Inputs + CNN Softmax	39.8	**0.29**
BIG LSTM+CNN Inputs + CNN Softmax + 128-dim correction	35.8	**0.39**
BIG LSTM+CNN Inputs + Char LSTM predictions	47.9	**0.23**

- CNN Softmax worse performance than normal softmax
- Correction factor improves performance
- Char LSTM predictions far below best performance
- Significantly less parameters for both models

Results

Model	Test Perplexity
Large Ensemble (Chelba et al., 2013)	43.8
RNN+KN-5 (Williams et al., 2015)	42.4
RNN+KN-5 (Ji et al., 2015a)	42.0
RNN+SNM10-skip (Shazeer et al., 2015)	41.3
Large Ensemble (Shazeer et al., 2015)	41.0
Our 10 best LSTM models (equal weights)	26.3
Our 10 best LSTM models (optimal weights)	26.1
10 LSTMs + KN-5 (equal weights)	25.3
10 LSTMs + KN-5 (optimal weights)	25.1
10 LSTMs + SNM10-skip (Shazeer et al., 2015)	**23.7**

Improvement from best n-gram

- Ensembling models very useful
- Contribution of best n-gram model minimal

Results

Table 3. The test perplexities of an LSTM-2048-512 trained with different losses versus number of epochs. The model needs about 40 minutes per epoch. First epoch is a bit slower because we slowly increase the number of workers.

EPOCHS	NCE	IS	TRAINING TIME [HOURS]
1	97	60	1
5	58	47.5	4
10	53	45	8
20	49	44	14
50	46.1	43.7	34

Results

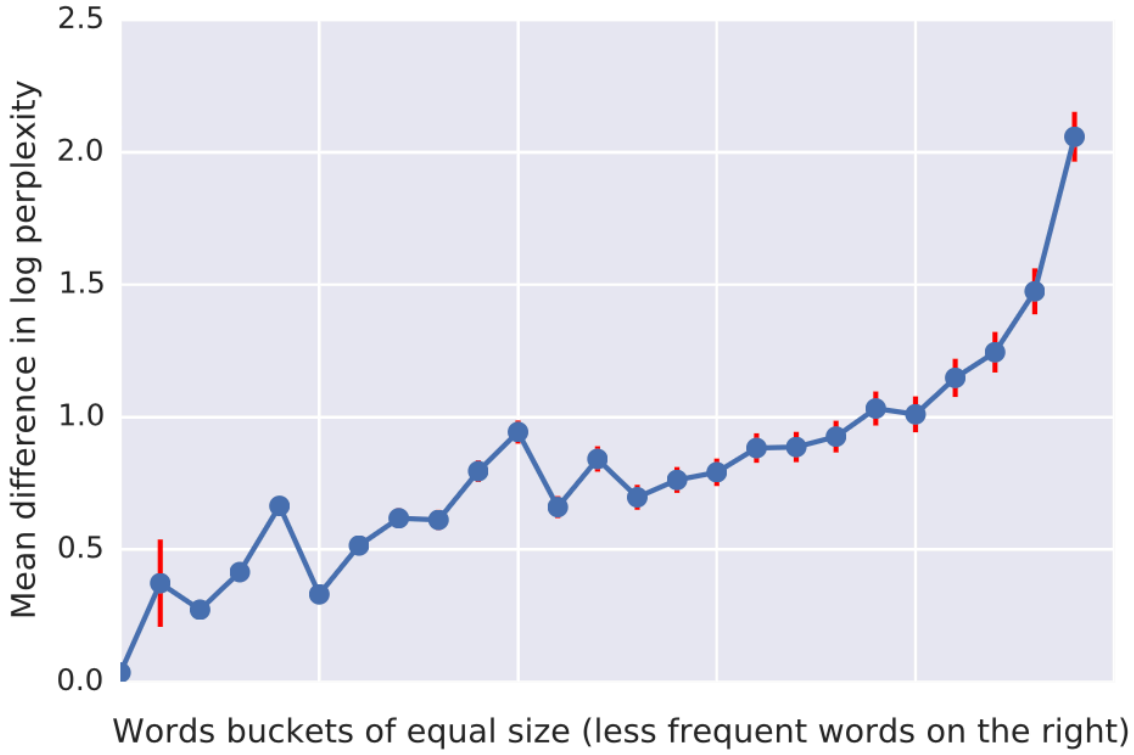


Figure 2. The difference in log probabilities between the best LSTM and KN-5 (higher is better). The words from the hold-out set are grouped into 25 buckets of equal size based on their frequencies.

Results - Conclusion

- Large RNN LMs trained on large amounts of data outperform n-gram models
- CNN character embeddings do not degrade performance while reducing model size
- Improvements from Interpolation with n-gram models only marginal

Examples

< S > With even more new technologies coming onto the market quickly during the past three years , an increasing number of companies now must tackle the ever-changing and ever-changing environmental challenges online . < S > Check back for updates on this breaking news story . < S > About 800 people gathered at Hever Castle on Long Beach from noon to 2pm , three to four times that of the funeral cort` ege . < S > We are aware of written instructions from the copyright holder not to , in any way , mention Rosenberg ’s negative comments if they are relevant as indicated in the documents , ” eBay said in a statement . < S > It is now known that coffee and cacao products can do no harm on the body . < S > Yuri Zhirkov was in attendance at the Stamford Bridge at the start of the second half but neither Drogba nor Malouda was able to push on through the Barcelona defence .

Commentary & Discussion

Commentary

- Cons:
  - Subword Segmentation
  - Attention-based approaches
- Pros:
  - Focus on larger datasets
  - Character-level approaches
  - Stacked architectures
  - Focus on model size optimization

Context

- Language Modeling (2016)
  - Task specific
  - Small datasets
  - Small models
  - N-grams

- This work:
  - Language modeling as an independent task
  - Large Datasets
  - Large Models
  - Superiority of RNNs

Paper gave direction for research in the future

Thank you!

Ernest Shackleton’s Terra Nova Expedition

Questions?

Appendix - Ensembling for LMs

- Ensembling: Combination of multiple models to improve performance
- Most commonly used method for LMs: Combination
  - Also called interpolation or just ensembling
- Linear interpolation: (Pusateri et al., 2019)

$$p^{\mathrm{LI}}(w|h) = \sum_i \lambda_i p_i(w|h)$$

  - Parameters optimized using standard optimization methods

Appendix - Perplexity

- Average per-word log-probability on the holdout data set

$$e^{-\frac{1}{N}\sum_i \ln p_{w_i}}$$

- Values between 1 and |V|, lower is better

https://huggingface.co/docs/transformers/perplexity

Appendix - BPE

- Subword segmentation:
  - First proposed by Sennrich et al., 2015 based on byte-pair encoding (BPE)
  - Dynamically generating vocabularies by merging frequent subwords until the desired vocabulary size is reached.
  - Able to fall back to character-level segmentation for OOV words but not suffering from long sequences as frequent words are part of the vocabulary.
  - Today standard (BERT using WordPiece, GPT using BPE)

Appendix - Cross Entropy Loss

- Usually in language modeling we train the system using the cross entropy loss between the true distribution of the target word and the probability of our system.

$$Loss(p^*, p) = -p^* \log(p) = -\sum_{i=1}^{|V|} p_i^* \log(p_i)$$

- Normally the target distribution is a one-hot-encoding of the target word. We therefore only need to compute the probability of our target word as for all other words the true distribution is 0.

$$Loss(p^*, p) = -\log(p_{y_t}) = -\log(p(y_t|y_{<t})).$$

https://lena-voita.github.io/nlp_course/language_modeling.html

Appendix - CNN embeddings

Table 4. Nearest neighbors in the character CNN embedding space of a few out-of-vocabulary words. Even for words that the model has never seen, the model usually still finds reasonable neighbors.

WORD	TOP-1	TOP-2	TOP-3
INCERDIBLE	INCREDIBLE	NONEDIBLE	EXTENDIBLE
WWW.A.COM	WWW.AA.COM	WWW.AAA.COM	WWW.CA.COM
7546	7646	7534	8566
TOWNHAL1	TOWNHALL	DJC2	MOODSWING360
KOMARSKI	KOHARSKI	KONARSKI	KOMANSKI

Importance Sampling Derivation

- Applying Bayes rule:

$$P(t_i = y|x_i, C_i) = P(t_i = y, C_i|x_i) / P(C_i|x_i)$$
$$= P(t_i = y|x_i)\, P(C_i|t_i = y, x_i) / P(C_i|x_i)$$
$$= P(y|x_i)\, P(C_i|t_i = y,\ x_i) / P(C_i|x_i)$$

- Now to compute $P(C_i|t_i = y, x_i)$, we note that in order for this to happen, S_i may or may not contain y, must contain all other elements of C_i, and must not contain any classes not in C_i. So:

$$P(t_i = y|x_i,\ C_i) = P(y|x_i) \prod_{y' \in C_i - \{y\}} Q(y'|x_i) \prod_{y' \in (L - C_i)} (1 - Q(y'|x_i)) \quad / P(C_i|x_i)$$

$$= \frac{P(y|x_i)}{Q(y|x_i)} \prod_{y' \in C_i} Q(y'|x_i) \prod_{y' \in (L - C_i)} (1 - Q(y'|x_i)) \quad / P(C_i|x_i)$$

$$= \frac{P(y|x_i)}{Q(y|x_i)} \quad / K(x_i, C_i)$$

https://www.tensorflow.org/extras/candidate_sampling.pdf

Importance Sampling Derivation

- K does not depend on y:

$$log(P(t_i = y|x_i, C_i)) = log(P(y|x_i)) - log(Q(y|x_i)) + K'(x_i, C_i)$$

$$Training\ Softmax\ Input = F(x,y) - log(Q(y|x)$$

https://www.tensorflow.org/extras/candidate_sampling.pdf