Advanced Natural Language Processing (ANLP)
Lecture 3: Representation Learning in NLP

Gabriel Stanovsky & **Roy Schwartz**

The Hebrew University of Jerusalem
roy.schwartz1@mail.huji.ac.il

THE HEBREW UNIVERSITY OF JERUSALEM

Recap

Part 1: What and How?

- Week 1: Intermediate applications
- Week 2: Downstream applications
- Week 3: How do we represent text?
- Week 4: Large language models! Or — which tools solve NLP tasks?

Motivation

Representing Language in Neural Networks

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language

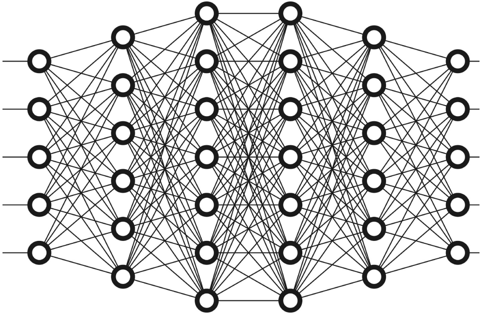
Lasagna

ארמון

To be or not to be, that is the question

أحضر الناس جوابا من لم يغضب

Как сказал Достоевский, «совершенно без надежды жить



Representation Learning

Turning Text into Vectors

$$\mathbf{v}_{\textsf{mountain}} = \begin{pmatrix} 0.23 \\ \text{-}0.21 \\ 0.15 \\ 0.61 \\ \vdots \\ 0.02 \\ \text{-}0.12 \end{pmatrix} \qquad \mathbf{v}_{\textsf{To be or not to be}} = \begin{pmatrix} 0.72 \\ 0.2 \\ 0.71 \\ 0.13 \\ \vdots \\ \text{-}0.1 \\ \text{-}0.11 \end{pmatrix}$$

Outline

Classic Word Representation Method (∼1970)

Word Embeddings (∼2010)

What is a Token?

Words in Context (∼2015)

Transformers (2017)

Pretrained Contextual Representation Models (2018)

$V_{1.0}$: One-Hot Vectors

- Given a vocabulary of size $|V|$, represent each word as an indicator vector $v \in \{0,1\}^{|V|}$
  - $v_{\text{cat}} = (1, 0, 0, \dots), v_{\text{dog}} = (0, 1, 0, 0, \dots), \dots$
- *The dog chased the cat*:

$$\begin{pmatrix} 0 & 0 & \cdots & \cdots & 1 & \cdots \\ 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & \cdots & 1 & \cdots & \cdots \\ 0 & 0 & \cdots & \cdots & 1 & \cdots \\ 0 & 1 & \cdots & \cdots & \cdots & 0 \end{pmatrix}$$

Problems

- No notion of similarity
  - v_{dog} is as similar to v_{cat} and v_{car}
- Vector size is huge
  - Typically dozens or hundreds of thousands

Distributional Semantics Hypothesis

Harris (1954)

Words that have similar contexts are likely to have similar meaning

$V_{2.0}$: Count Models

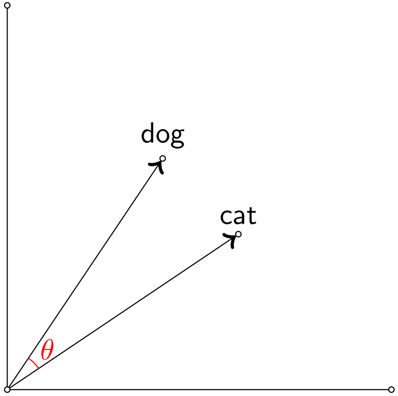
Aka Vector-space Models (Salton, 1971)

- Each element $\mathbf{v}_{w_i} \in \mathbf{v}_w$ represents the co-occurrence of w with another word i
  - $\mathbf{v}_{\text{dog}} = $ (cat: 10, leash: 15, loyal: 27, bone: 8, piano: 0, cloud: 0, ...)
- Vector dimension is still very large (vocabulary size)
  - Though vectors are typically sparse
- But vectors now have some notion of *similarity*

Count Models

Example

$$\mathbf{v}_{\mathsf{dog}} = \begin{pmatrix} 0 \\ 0 \\ 15 \\ 17 \\ \vdots \\ 0 \\ 102 \end{pmatrix}, \mathbf{v}_{\mathsf{cat}} = \begin{pmatrix} 0 \\ 2 \\ 11 \\ 13 \\ \vdots \\ 20 \\ 11 \end{pmatrix}$$



Variants of Count Models

- Reduce the effect of high frequency words by applying a weighting scheme
  - Pointwise mutual information (PMI), TF-IDF
- Smoothing by dimensionality reduction
  - Singular value decomposition (SVD), principal component analysis (PCA), matrix factorization methods
  - Vector size is much smaller (typically in the hundreds)
- A popular approach until the early 2010s
  - Turney and Pantel (2010); Clark (2015)

Outline

- Classic Word Representation Method ($\sim$1970)
- Word Embeddings ($\sim$2010)
- What is a Token?
- Words in Context ($\sim$2015)
- Transformers (2017)
- Pretrained Contextual Representation Models (2018)

- Classic Word Representation Method ($\sim$1970)
- Word Embeddings ($\sim$2010)
- What is a Token?
- Words in Context ($\sim$2015)
- Transformers (2017)
- Pretrained Contextual Representation Models (2018)

$V_{3.0}$: Predict Models

Aka Word Embeddings

- A new generation of vector space models
- Train a supervised machine learning algorithm to predict $p(\text{word}|\text{context})$
- Models learn a latent vector representation of each word
  - These representations turn out to be quite effective vector space representations

Word Embeddings

- Vector size is typically a *few dozens* to a *few hundreds*
- Vector elements are generally **uninterpretable**
- Developed to initialize feature vectors in deep learning models
  - Initially language models, later virtually every sequence level NLP task

word2vec

Mikolov et al. (2013)

- A software toolkit for running various word embedding algorithms
- Continuous bag-of-words: $\underset{\theta}{\operatorname{argmax}} \prod_{w \in \text{corpus}} p(w|C(w); \theta)$
- Skip-gram: $\underset{\theta}{\operatorname{argmax}} \prod_{(w,c) \in \text{corpus}} p(c|w; \theta)$
  - Where: $p(c|w;\theta) = \frac{e^{v_c} \cdot e^{v_w}}{\sum_{c' \in C} e^{v_c} \cdot e^{v_w}}$
- Problem: the denominator is expensive to compute

Solution 1: Did this Pair Come from the Training Data?

- Instead of computing $p(c|w;\theta)$, define D to be whether c is a real context of w
- $p(D=1|w,c,\theta) = \frac{1}{1+e^{-v_c \cdot v_w}}$
- And maximize

$$\sum_{(w,c)\in D} p(D=1|w,c,\theta)$$

- Problem: this has a trivial solution
  - Set θ such that $\forall_{v_c,v_w} : v_c = v_w$ and $v_c \cdot v_w = K$ for some large K

Based on Goldberg and Levy (2014)

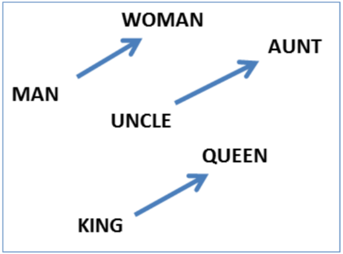
Solution 2: Negative Sampling

- Randomly sample **negative** (*word,context*) pairs D'
- $p(D=0|w,c,\theta) = 1 - p(D=1|w,c,\theta)$
- maximize
$$\sum_{(w,c)\in D} p(D=1|w,c,\theta) + \sum_{(w,c)\in D'} p(D=0|w,c,\theta)$$

Based on Goldberg and Levy (2014)

Skip-gram with Negative Sampling (SGNS)

- Obtained significant improvements on a range of lexical semantic tasks
- Is very fast to train, even on large corpora
- *Emerging* properties



Count vs. Predict

- Don't count, Predict! (Baroni et al., 2014)
- But...
  Neural embeddings are implicitly matrix factorization tools (Levy and Goldberg, 2014)
- So?...
  It's all about *hyper-parameters* (Levy et al., 2015)
- The bottom line:
  word2vec is a very good *implementation*

Scale

- count Brown Clusters (Brown et al., 1992)
  - A classical hierarchical clustering of words based on their contexts
  - A pre-trained version trained on 43M words, vocabulary of 280K (Liang, 2005)
  - Trained for 3 days
- predict Collobert et al. (2011)
  - One of the early word embedding works
  - Trained on 850M words (vocabulary size of 130K)
  - Trained for 7 weeks
- predict word2vec (Mikolov et al., 2013)
  - Largest model trained on 6B words (vocabulary of 1M words) for 3 days
- predict GloVe (Pennington et al., 2014)
  - Trained on 840B words (vocabulary 2.2M words)

Outline

Classic Word Representation Method (∼1970)

Word Embeddings (∼2010)

What is a Token?

Words in Context (∼2015)

Transformers (2017)

Pretrained Contextual Representation Models (2018)

Words as the Basic Representation Units

- The basic representation approach maps each word to its own vector
- This ignores the morphological structure of words
  - v_{opens} should be close to v_{opening}
- As well as typos
  - v_{opennig} should be close to v_{opening}
- These are often hard to learn from text

Other Representation Units

- Character embeddings
  - Learn vectors for each character rather than words
  - Useful in, e.g., Machine translation (Ling et al., 2015), Syntactic parsing (Ballesteros et al., 2015)
- Fixed-size character n-grams (Neubig et al., 2013; Schütze, 2017)
- **Byte-pair encoding**

Byte-Pair Encoding (BPE)

AKA word-pieces (Gage, 1994)

- Identify the most frequent (**varied length**) character n-grams
- Procedure: iteratively merge the most frequent bigrams in the corpus
- Toy example:
  - cafabdfacdfac
  - Z = fa → caZbdZcdZc
  - Y=Zc (=fac) → caZbdYdY
  - New vocabulary: a, b, c, d, fa, fac
- Identifies frequent words, prefices, suffices and infices
  - Infrequent ngrams are represented using more than one token
- Many of todays networks are based on this technology
- Not grounded in any linguistic principle

Additional Knowledge

- Enhance vectors with external knowledge source
  - E.g., **dictionaries**, **thesauri**
- Combination of textual and perceptual representations (multimodal embeddings)
  - Most prominently **visual**
- Mapping embeddings in **different languages** into the same space (multilingual embeddings)
  - $\mathbf{v}_{\text{dog}} \sim \mathbf{v}_{\text{perro}}$
  - Useful for multi-lingual tasks, as well as low-resource scenarios
  - Most approaches use bilingual dictionaries or parallel corpora

Outline

Words can have Multiple Meanings

- *What is your* **date** *of birth?*

- **Date** *is my favorite fruit*

- *Mary took John out on a* **date**

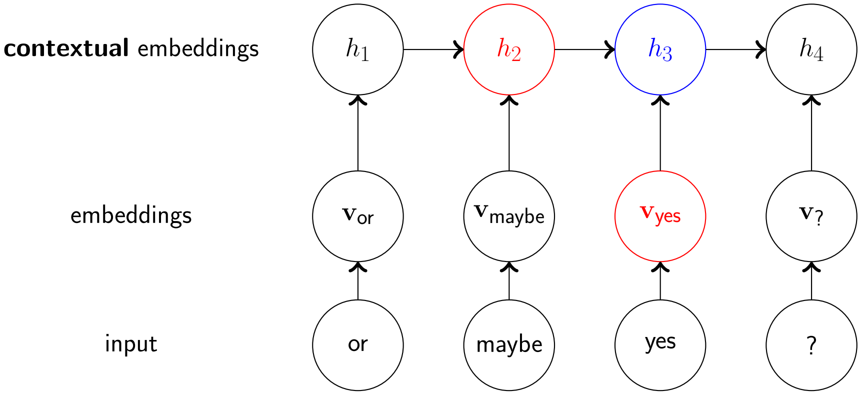
- But word embeddings assign each token a *single vector*

Contextual Embeddings

- The representation of a token is a function of its neighbors **in this document**
- A building block of most NLP neural networks

Recurrent Neural Networks (RNNs)

Elman (1990)



RNN Variants

Elman (vanilla) RNN (Elman, 1990)

- $h_t = \sigma(Wx_t + Uh_{t-1} + b)$
  - W, U are learned parameter matrices, b is a learned bias term vector

RNN Variants

LSTM (Hochreiter and Schmidhuber, 1997)

- Adds *gates* to the vanilla RNN
- *Forget* gate (f_t) controls how much we "forget" the past, *input* gate (i_t) controls how much we rely on the current token

$$
\boxed{\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c)
\end{aligned}}
$$

$$
\boxed{c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t}
$$

$$
h_t = o_t \circ \sigma_h(c_t)
$$

RNN Variants

GRU (Cho et al., 2014)

- Merges input and forget gates ($f_t = 1 - i_t$)

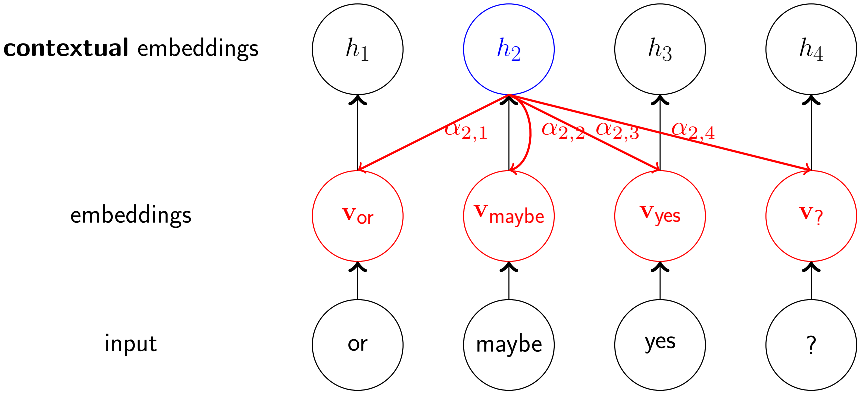
$$
\begin{aligned}
z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \\
\hat{h}_t &= \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \\
&\boxed{h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t}
\end{aligned}
$$

Self-Attention

- Self-attention can be thought of as a weighted sum of all the other token-vectors in the document
  - $h_t = \sum_i \alpha_{t,i} x_i$
  - The $\alpha_{t,i}$ weights are learned
- Unlike RNNs, can be efficiently **parallelized**
- The main component in the **Transformer** model (Vaswani et al., 2017)

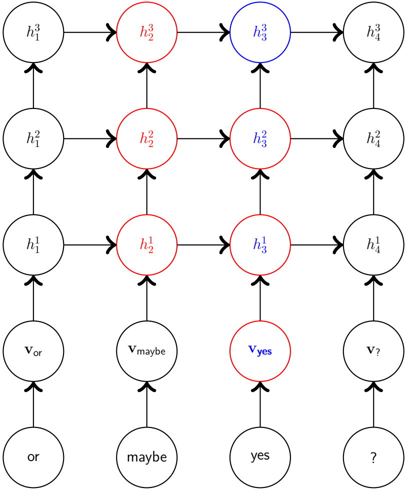
Self-Attention

Example



More on Contextual Embeddings

- Typically composed of multiple layers
- Learned end-to-end (task-specific)
  - Require large amounts of labeled data



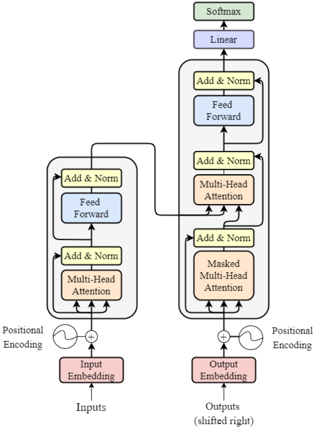
Outline

Classic Word Representation Method (∼1970)
Word Embeddings (∼2010)
What is a Token?
Words in Context (∼2015)
Transformers (2017)
Pretrained Contextual Representation Models (2018)

Transformer

Vaswani et al. (2017)

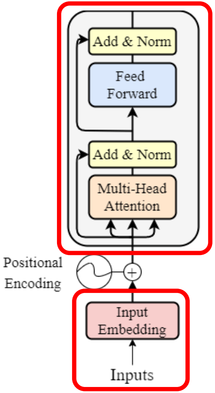
- The most dominant model in NLP
  - And vision, speech, computational biology, computational chemistry, and more



Transformers – Step by Step

Input

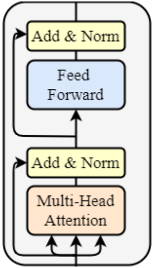
- Words are first embedded into context-independent representation
  - Typically using BPE
- This is fed to the transformer block



Transformers – Step by Step

Transformer Block

- The Transformer block is composed of two (three?) parts
- Multi-head attention
- Feed-forward layer
- Residual connection and layer normalization



Attention in Transformers

- Self-attention represents each word using a weighing of the tokens
- This weighing is parameterized by three matrices of size:

$$W_{k(eys)}, W_{q(ueries)}, W_{V(alues)}$$

- Each matrix is multiplied by the input matrix X, resulting in $n \times D$ matrices:

$$K = X \times W_k \qquad Q = X \times W_q \qquad V = X \times W_v$$

- The attention weights are then computed (an $n \times n$ matrix):

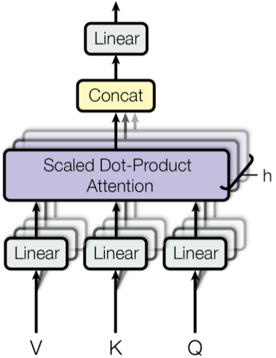
$$A = softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right)$$

- And the output of the attention component is:

$$Z = A \cdot V$$

Multi-Headed Attention

- We might want to consider different weighings
- Multi-head attention allows learning them
  - Implemented via different W_k, W_q, W_V matrices for each head
- They are then concatenated and passed into a linear projection to form a single representation



Feed-forward Layer

- The self-attention output is fed into a feed-forward layer

$$FFN(x) = ReLU(xW_1 + b_1)W_2 + b_2$$

- Where W's and b's are learned

Add & Norm

- Residual Connections are a mechanism for allowing the model to "skip" over a given layer if it wants to
  - Aka, Highway/Skip Connections; He et al. (2016)
- They are followed by layer normalization (Ba et al., 2016)

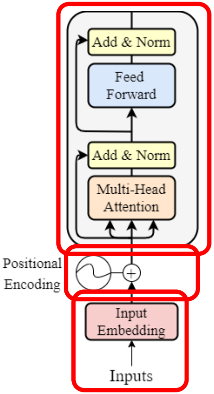
$$LayerNorm(x) = \frac{x - mean}{std + \epsilon}$$

$$x' = LayerNorm(x + SubLayer(x))$$

Positional Encoding

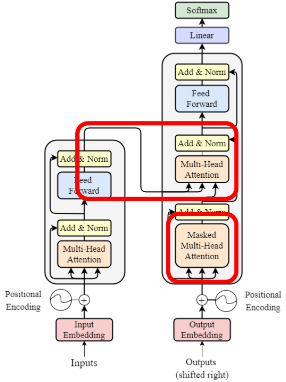
- Unlike RNNs, Transformers do not have a builtin mechanism for dealing with the order of sequences
- An encoding of the position of each word is added to the input embeddings
- This allows the model to use the positional information when representing the document
- The original Transformer model used sine and cosine functions of different frequencies
  - This is chosen as it should allow the model to easily learn to attend by relative positions, since for any position i and fixed offset k, $f(i+k)$ can be represented as a linear function of $f(i)$
  - Other, more advanced methods exists (Press et al., 2022)

Wrapping it all up



Encoder-Decoder

- Transformers can be used as seq2seq models
- What we described so far is the Encoder architecture
- The decoder is very similar, except it has two attention components:
- Masked attention, which attends the previously generated tokens
- Cross attention, which attends the input



Outline

Classic Word Representation Method (∼1970)

Word Embeddings (∼2010)

What is a Token?

Words in Context (∼2015)

Transformers (2017)

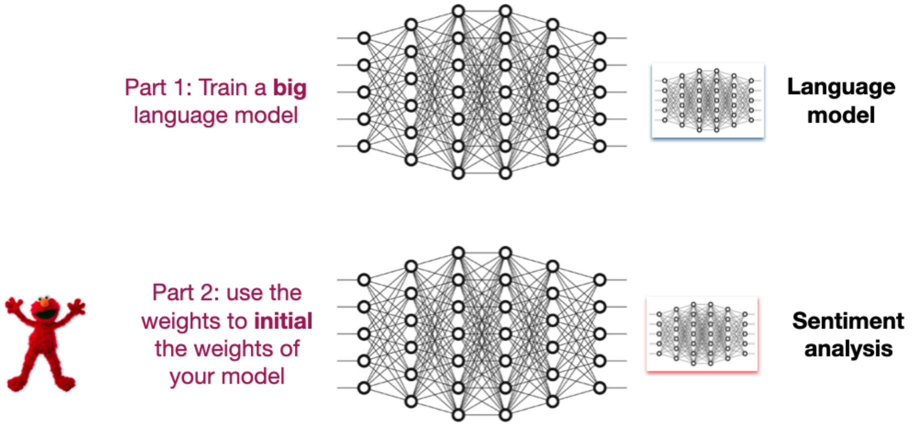
Pretrained Contextual Representation Models (2018)

Can we Pre-Train Contextual Embeddings?

- Much like type embeddings, contextual embeddings are also shared between tasks
  - They can also be learned from plain text!
- Pre-train a deep language model with contextual embeddings
  - Predict the next word given the document prefix (e.g., *The boy ate a [MASK]*)
- Use the learned weights to initialize a new network
  - Replacing only the classification layer

ELMo

Peters et al. (2018)



ELMo

Peters et al. (2018)

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

A new Generation of NLP

- End-to-end $\rightarrow$ pre-train and fine-tune
- A wealth of pretrained contextual models
  - GPT (Radford et al., 2018)
  - BERT (Devlin et al., 2019)
  - RoBERTa (Liu et al., 2019)
  - GPT-2 (Radford et al., 2019)
  - T5 (Raffel et al., 2020)
  - GPT-3 (Brown et al., 2020)
  - PaLM (Chowdhery et al., 2022)
- New state-of-the-art results in virtually every single NLP task

Transformer-based Pretrained Contextual Embeddings

- ELMo was based on LSTMs
- Following, virtually all other pretrained contextual embeddings are **Transformer**-based
- Are LSTMs dead?
  - Kind of, or maybe not (Merity, 2019)?

Models are getting Bigger

References I

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. arXiv:1607.06450.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proc. of EMNLP*, 2015.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proc. of ACL*, 2014.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):467–480, 1992. URL `https://www.aclweb.org/anthology/J92-4003`.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krüger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proc. of NeurIPS*, 2020.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proc. of SSST*, 2014.

References II

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling language modeling with pathways, 2022. URL `https://arxiv.org/abs/2204.02311`. arXiv:2204.02311.

Stephen Clark. *Vector Space Models of Lexical Meaning*, chapter 16, pages 493–522. John Wiley & Sons, Ltd, 2015. ISBN 9781118882139. doi: https://doi.org/10.1002/9781118882139.ch16. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118882139.ch16`.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*, 2019. doi: 10.18653/v1/N19-1423. URL `https://www.aclweb.org/anthology/N19-1423`.

References III

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

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

Yoav Goldberg and Omer Levy. word2vec explained: Deriving mikolov et al.?s negative-sampling word-embedding method, 2014. arXiv:1402.3722.

Zelig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Omer Levy and Yoav Goldberg. Neural word embeddings as implicit matrix factorization. In *Proc. of NeurIPS*, 2014.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3:211–225, 2015.

Percy Liang. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology, 2005.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. Character-based neural machine translation, 2015. arXiv:1511.04586.

References IV

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

Stephen Merity. Single headed attention RNN: Stop thinking with your head, 2019. arXiv:1911.11423.

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

Graham Neubig, Taro Watanabe, Shinsuke Mori, and Tatsuya Kawahara. Substring-based machine translation. *Machine Translation*, 27(2):139–166, 2013.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proc. of EMNLP*, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *Proc. of ICLR*, 2022. URL `https://openreview.net/forum?id=R8sQPpGCv0`.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

References V

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020. URL `http://jmlr.org/papers/v21/20-074.html`.

Gerard Salton. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

Hinrich Schütze. Nonsymbolic text representation. In *Proc. of EACL*, 2017.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Communications of the ACM (CACM)*, 63(12):54–63, November 2020. ISSN 0001-0782. doi: 10.1145/3381831. URL `https://doi.org/10.1145/3381831`.

Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training NLP models: A concise overview, 2020. arXiv:2004.08900.

Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence research*, 37(1):141–188, 2010.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of NeurIPS*, 2017.