Artificial Intelligence for Business Research @Antai

Deep-Learning-based NLP

Renyu (Philip) Zhang

1

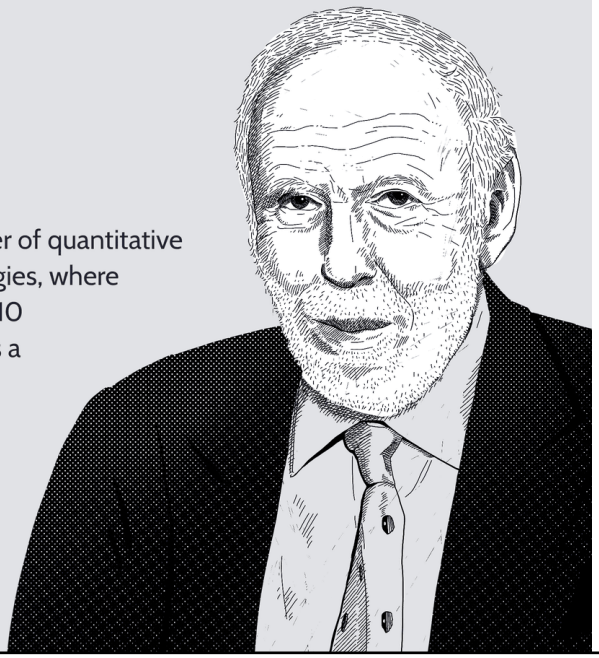
Jim Simons

Born: April 25, 1938
Died: May 10, 2024

Investor

- Known as the "Quant King," founder of quantitative hedge fund Renaissance Technologies, where he served as CEO from 1982 to 2010
- Before his career in finance, he was a master code breaker for the IDA, a successful mathematician, and a professor at MIT and Harvard
- Served as the chair of the mathematics department at Stony Brook University

Investopedia



2

2

Text as Data

Journal of Economic Literature 2019, 57(3), 535–574
https://doi.org/10.1257/jel.20181020

Text as Data†

Matthew Gentzkow, Bryan Kelly, and Matt Taddy*

An ever-increasing share of human interaction, communication, and culture is recorded as digital text. We provide an introduction to the use of text as an input to economic research. We discuss the features that make text different from other forms of data, offer a practical overview of relevant statistical methods, and survey a variety of applications. (JEL C38, C55, L82, Z13)

0. Pre-processing;

1. Represent raw text $\mathcal{D}$ as a numerical array $\mathbf{C}$;
2. Map $\mathbf{C}$ to predicted values $\hat{\mathbf{V}}$ of unknown outcomes $\mathbf{V}$; and
3. Use $\hat{\mathbf{V}}$ in subsequent descriptive or causal analysis.

3

3

NLP Roadmap

Attention is all you need
A Vaswani, N Shazeer, N Parmar… - Advances in neural …, 2017 - proceedings.neurips.cc
… the number of **attention** heads and the **attention** key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head **attention** is 0.9 …
★ Save 99 Cite Cited by 36430 Related articles All 35 versions »

Bert: Pre-training of deep bidirectional transformers for language understanding
J Devlin, MW Chang, K Lee, K Toutanova - arXiv preprint arXiv …, 2018 - arxiv.org
… We introduce **BERT** and its detailed implementation in this … For finetuning, the **BERT** model is first initialized with the pre-trained … , we present **BERT** fine-tuning results on 11 **NLP** tasks. …
☆ Save 99 Cite Cited by 34042 Related articles All 41 versions »

Probability & Vector Representations of NLP → Word2vec & Seq2seq → Transformer-based Models → Large Language Models (LLM)

Before 2010 | 2013-2017 | 2018-2022 | 2023-now

Efficient estimation of word representations in vector space
T Mikolov, K Chen, G Corrado, J Dean - arXiv preprint arXiv:1301.3781, 2013 - arxiv.org
… **vector** X = **vector**("biggest")−**vector**("big") + **vector**("small"). Then, we search in the **vector space** for the **word** … question (we discard the input question **words** during this search). When the …
☆ Save 99 Cite Cited by 27227 Related articles All 48 versions »

[PDF] **Glove: Global vectors** for word representation
J Pennington, R Socher … - Proceedings of the 2014 …, 2014 - aclanthology.org
… We use our insights to construct a new model for word representation which we call **GloVe**, for **Global Vectors**, because the **global** corpus statistics are captured directly by the model. …
☆ Save 99 Cite Cited by 26265 Related articles All 34 versions »

[PDF] Language models are unsupervised multitask learners
A Radford, J Wu, R Child, D Luan, D Amodei … - OpenAI blog, 2019 - persagen.com
Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on taskspecific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset-matching or exceeding
☆ Save 99 Cite Cited by 2834 Related articles All 16 versions »

Training language models to follow instructions with human feedback
L Ouyang, J Wu, X Jiang, D Almeida… - Advances in …, 2022 - proceedings.neurips.cc
Making language models bigger does not inherently make them better at following a user's intent. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. In other words, these models are not aligned with their users. In this paper, we show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning with human feedback. Starting with a set of labeler-written prompts and prompts submitted through a language model API, we collect a dataset of …
☆ Save 99 Cite Cited by 4049 Related articles All 10 versions »

4

4

Agenda

- Word2Vec: Continuous Bag of Words (CBOW), Skip-Gram, Evaluations, Applications
- RNN, LSTM, Seq2Seq, and Applications
- Attention Mechanism and Transformer

5

5

Word2Vec: Continuous Bag of Words (CBOW)

- References:
  - https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture01-wordvecs1-public.pdf
  - https://web.stanford.edu/class/cs224n/readings/cs224n_winter2023_lecture1_notes_draft.pdf
- So far, our word representations are based on **word-document co-occurrence matrices**.
- Word embedding is a way to obtain word representations via prediction, called **self-supervised learning**.
- Distributional semantics: A word's meaning is provided by the words that **frequently appear close-by**.
- Context of a word w: The set of words that appear **nearby with a fixed window size**.

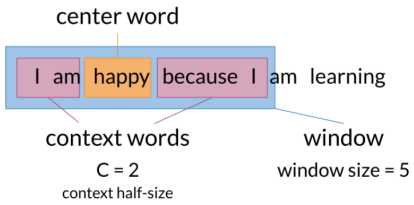
…government debt problems turning into ***banking*** *crises as happened in 2009…*
…saying that Europe needs unified ***banking*** *regulation to replace the hodgepodge…*
…India has just given its ***banking*** *system a shot in the arm…*

These context words will represent ***banking***

6

Word2Vec: Continuous Bag of Words (CBOW)

- References:
  - https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture01-wordvecs1-public.pdf
  - https://web.stanford.edu/class/cs224n/readings/cs224n_winter2023_lecture1_notes_draft.pdf
- Continuous Bag of Words (CBOW): Use the outside words (o) to predict the center word (c).
- Skip-gram: Use the **center word (c)** to predict the distribution of **outside words (o)**.



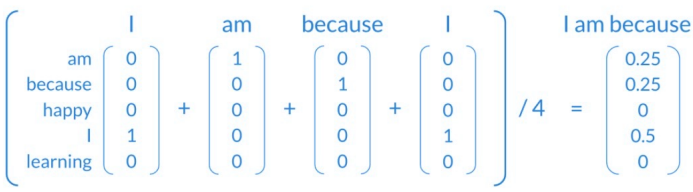
$$\begin{aligned}\text{minimize } J &= -\log P(w_c|w_{c-m},\ldots,w_{c-1},w_{c+1},\ldots,w_{c+m}) \\ &= -\log P(u_c|\hat{v}) \\ &= -\log \frac{\exp(u_c^T\hat{v})}{\sum_{j=1}^{|V|}\exp(u_j^T\hat{v})} \\ &= -u_c^T\hat{v} + \log\sum_{j=1}^{|V|}\exp(u_j^T\hat{v})\end{aligned}$$

7

7

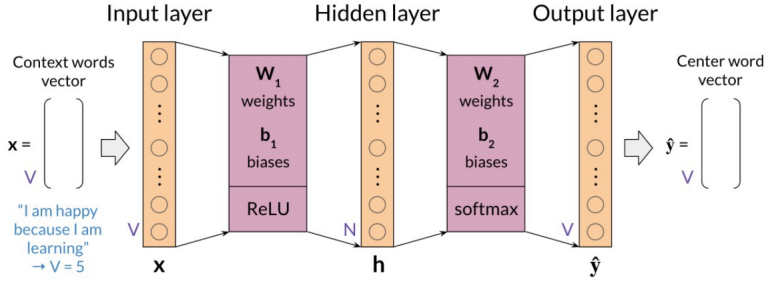
CBOW: Deep Learning Architecture

- References:
  - https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture01-wordvecs1-public.pdf
  - https://web.stanford.edu/class/cs224n/readings/cs224n_winter2023_lecture1_notes_draft.pdf



I am happy because I am learning.

Center Word (c)



Label: Center Word 1-hot.

Loss: Cross entropy.

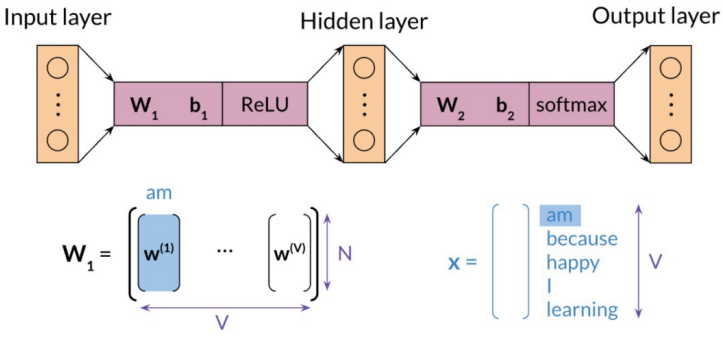
Use **backpropagation** to estimate the weights and biases.

8

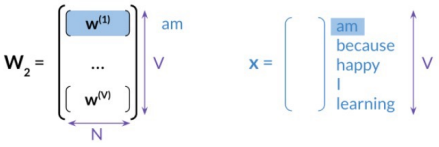
8

CBOW: Everything Together

1. Convert the text into training data (center word, outside words).
2. Create a 3-layer NN (Dense + ReLU + Softmax)
3. Use back propagation to train the model.
4. Extract the word embeddings with the model.
5. Test the performance of the word embeddings.



If you were to use w_1, each column will correspond to the embeddings of a specific word. You can also use w_2 as follows:



9

More on CBOW

- Original code: https://github.com/tmikolov/word2vec
- Trained on Google News Corpus with 6B Tokens (~10Bb)
  - The widely used public data set wikiText 103 has 100M tokens.
  - https://pytorch.org/text/stable/datasets.html#id16
- Trained vectors can be found here:
  - https://github.com/mmihaltz/word2vec-GoogleNews-vectors
- Required computational power:

Efficient estimation of word representations in vector space
T Mikolov, K Chen, G Corrado, J Dean - arXiv preprint arXiv:1301.3781, 2013 - arxiv.org
... **vector** X = **vector**("biggest")−**vector**("big") + **vector**("small"). Then, we search in the **vector space** for the **word** ... question (we discard the input question **words** during this search). When the ...
☆ Save 99 Cite Cited by 28876 Related articles All 48 versions »

Distributed representations of words and phrases and their compositionality
T Mikolov, I Sutskever, K Chen ... - Advances in neural ..., 2013 - proceedings.neurips.cc
The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. In this paper we present several improvements that make the Skip-gram model more expressive and enable it to learn higher quality vectors more rapidly. We show that by subsampling frequent words we obtain significant speedup, and also learn higher quality representations as measured by our tasks. We also introduce ...
☆ Save 99 Cite Cited by 33549 Related articles All 56 versions »

NeurIPS 2023 Test-of-Time Award

Table 6: *Comparison of models trained using the DistBelief distributed framework. Note that training of NNLM with 1000-dimensional vectors would take too long to complete.*

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

Table 2: *Accuracy on subset of the Semantic-Syntactic Word Relationship test set, using word vectors from the CBOW architecture with limited vocabulary. Only questions containing words from the most frequent 30k words are used.*

Dimensionality / Training words	24M	49M	98M	196M	391M	783M
50	13.4	15.7	18.6	19.1	22.5	23.2
100	19.4	23.1	27.8	28.7	33.4	32.2
300	23.2	29.2	35.3	38.6	43.7	45.9
600	24.0	30.1	36.5	40.8	46.6	50.4

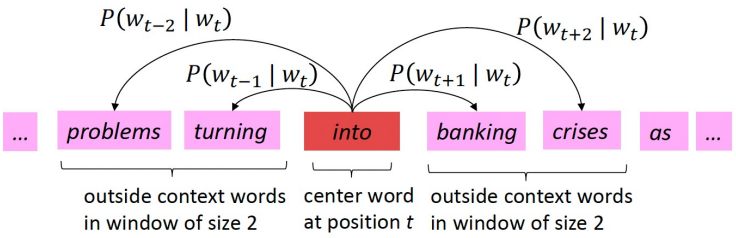
Document → Embedding → Taking Mean → Classification → Sentiment

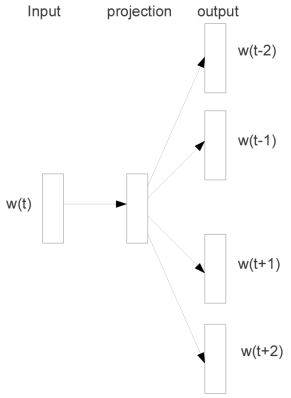
10

Word2Vec: Skip-Gram

- References:
  - https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture01-wordvecs1-public.pdf
  - https://web.stanford.edu/class/cs224n/readings/cs224n_winter2023_lecture1_notes_draft.pdf
- Skip-gram: Like CBOW but the inverse, using the center word to predict the outside words.

Example windows and process for computing $P(w_{t+j} \mid w_t)$



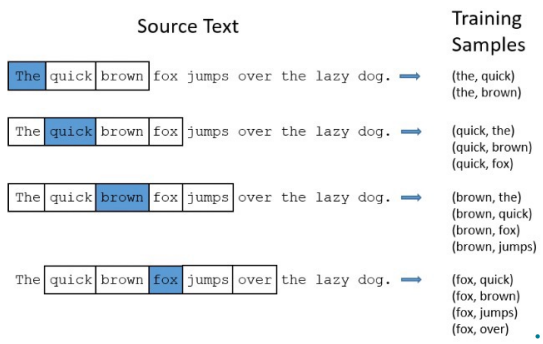


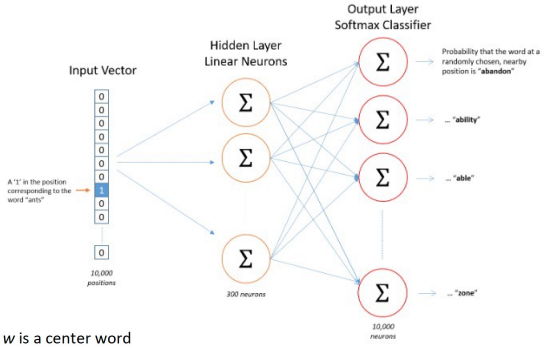
11

11

Word2Vec: Skip-Gram

- References:
  - https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture01-wordvecs1-public.pdf
  - https://web.stanford.edu/class/cs224n/readings/cs224n_winter2023_lecture1_notes_draft.pdf





We want to minimize the objective function:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\le j\le m \\ j\neq 0}} \log P(w_{t+j} \mid w_t;\theta)$$

- v_w when w is a center word
- u_w when w is a context word

Then for a center word c and a context word o:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}$$

12

12

Negative Sampling

- References:
  - https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture02-wordvecs2.pdf
  - https://web.stanford.edu/class/cs224n/readings/cs224n_winter2023_lecture1_notes_draft.pdf

- **Negative Sampling**: For **a positive case (numerator)**, we only use **a small set of negative cases (denominator)** to update the weights, saving a lot of computations.
  - **Randomly select just a small number of "negative" words** (let's say 5) to update the weights for.

- **Question**: Your embedding size is 300 and you have 10,000 unique words, and you train on (quick, fox):
  - If you don't use negative sampling, how many weights are you updating?
  - If you use 5-word negative sampling, how many weights are you updating?

13

13

How to Evaluate Language Models?

- Reference: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture02-wordvecs2.pdf

- **Intrinsic Evaluation:**
  - Measure the quality of a model on a **specific/intermediate task independent of any application**.
  - For example, an intrinsic evaluation is analogy test. It allows you to capture semantic **analogies** as, "France" is to "Paris" as "Italy" is to <?> and syntactic analogies as "seen" is to "saw" as "been" is to <?>.
  - Fast to compute, helpful to understand the system, but unclear how it correlates with real tasks.
  - Analogies, similarities, etc.

- **Extrinsic Evaluation:**
  - The best way to evaluate the performance of a language model is to embed it in an application and **measure how much the application improves**.
  - **Computationally costly**.
  - Unclear which part of the system (the subsystem or its interactions to other subsystems) should the performance changes be attributed to.
  - **Name Entity Recognition (NER)**, etc.

14

14

Intrinsic Evaluation: Word Analogy

- Word analogy task:

When given a pair of words a and a^* and a third word b, the analogy relationship between a and a^* can be used to find the corresponding word b^* to b. Mathematically, it is expressed as

$$a : a^* :: b : \_\_, \quad (7)$$

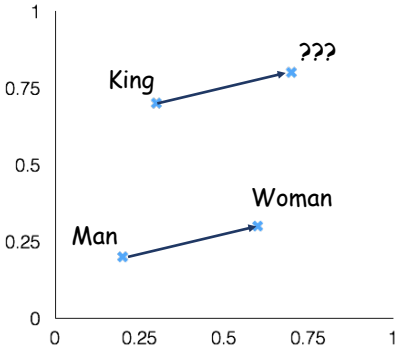
where the blank is b^*. One example could be

$$\text{write} : \text{writing} :: \text{read} : \underline{\text{reading}}. \quad (8)$$

The 3CosAdd method [33] solves for b^* using the following equation:

$$b^* = \underset{b'}{\text{argmax}}(\cos(b', a^* - a + b)), \quad (9)$$

	Word Analogy Datasets							
	Google		Semantic		Syntactic		MSR	
	Add	Mul	Add	Mul	Add	Mul	Add	Mul
SGNS	**71.8**	**73.4**	**77.6**	**78.1**	67.1	**69.5**	**56.7**	**59.7**
CBOW	70.7	70.8	74.4	74.1	**67.6**	68.1	56.2	56.8
GloVe	68.4	68.7	76.1	75.9	61.9	62.7	50.3	51.6
FastText	40.5	45.1	19.1	24.8	58.3	61.9	48.6	52.2
ngram2vec	70.1	71.3	75.7	75.7	65.3	67.6	53.8	56.6
Dict2vec	48.5	50.5	45.1	47.4	51.4	53.1	36.5	38.9



- You should remove the input vectors from the search of b'.
- Datasets: https://aclweb.org/aclwiki/Google_analogy_test_set_(State_of_the_art)

15

15

Intrinsic Evaluation: Word Similarity

- Given two representations of two words, we can compute their similarities:

$$\cos(w_x, w_y) = \frac{w_x \cdot w_y}{||w_x||\ ||w_y||},$$

- Example dataset: WordSim353: https://www.kaggle.com/datasets/julianschelb/wordsim353-crowd

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

- The following is a summary of word similarity test datasets:

TABLE I
WORD SIMILARITY DATASETS USED IN OUR EXPERIMENTS WHERE PAIRS INDICATE THE NUMBER OF WORD PAIRS IN EACH DATASET.

Name	Pairs	Year
WS-353 [40]	353	2002
WS-353-SIM [41]	203	2009
WS-353-REL [41]	252	2009
MC-30 [42]	30	1991
RG-65 [43]	65	1965
Rare-Word (RW) [44]	2034	2013
MEN [45]	3000	2012
MTurk-287 [46]	287	2011
MTurk-771 [47]	771	2012
YP-130 [48]	130	2006
SimLex-999 [49]	999	2014
Verb-143 [50]	143	2014
SimVerb-3500 [51]	3500	2016

Evaluating word embedding models: methods and experimental results
B Wang, A Wang, F Chen, Y Wang… - APSIPA transactions on …, 2019 - cambridge.org
Extensive evaluation on a large number of word embedding models for language processing applications is conducted in this work. First, we introduce popular word embedding models and discuss desired properties of word models and evaluation methods (or evaluators). Then, we categorize evaluators into intrinsic and extrinsic two types. Intrinsic evaluators test the quality of a representation independent of specific natural language processing tasks while extrinsic evaluators use word embeddings as input features to a …
☆ Save 99 Cite Cited by 188 Related articles All 6 versions »

Model	Size	WS353	MC	RG	SCWS	RW
SVD	6B	35.3	35.1	42.5	38.3	25.6
SVD-S	6B	56.5	71.5	71.0	53.6	34.7
SVD-L	6B	65.7	72.7	75.1	56.5	37.0
CBOW†	6B	57.2	65.6	68.2	57.0	32.5
SG†	6B	62.8	65.2	69.7	58.1	37.2
GloVe	6B	65.8	72.7	77.8	53.9	38.1
SVD-L	42B	74.0	76.4	74.1	58.3	39.9
GloVe	42B	**75.9**	**83.6**	**82.9**	**59.6**	**47.8**
CBOW*	100B	68.4	79.6	75.4	59.4	45.5

16

16

Extrinsic Evaluation: Named Entity Recognition

- Named Entity Recognition (NER): Identify the references to a person, location, organization, time, etc. https://en.wikipedia.org/wiki/Named-entity_recognition

Philip lives in Shenzhen and Shanghai in 2024.

Person Location Time

- Use **window classification** with a logistic classifier for NER.
- Classify each human-labeled word using the neighboring words in its context window.

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

Example: Classify "Paris" as +/– location in context of sentence with window length 2:

the museums in Paris are amazing to see .

$X_{window} = [\ x_{museums}\ \ x_{in}\ \ x_{Paris}\ \ x_{are}\ \ x_{amazing}\ ]^T$

17

17

Application: Text Algorithms in Economics

Annual Review of Economics

Text Algorithms in Economics

Elliott Ash[1] and Stephen Hansen[2,3]

[1]Center for Law and Economics, ETH Zurich, Zurich, Switzerland
[2]Department of Economics, University College London, London, United Kingdom; email: stephen.hansen@ucl.ac.uk
[3]Centre for Economic Policy Research, London, United Kingdom

Keywords

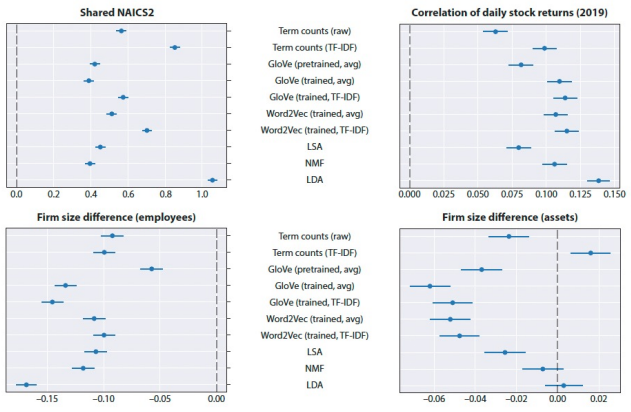
text as data, topic models, word embeddings, large language models, transformer models

Abstract

This article provides an overview of the methods used for algorithmic text analysis in economics, with a focus on three key contributions. First, we introduce methods for representing documents as high-dimensional count vectors over vocabulary terms, for representing words as vectors, and for representing word sequences as embedding vectors. Second, we define four core empirical tasks that encompass most text-as-data research in economics and enumerate the various approaches that have been taken so far to accomplish these tasks. Finally, we flag limitations in the current literature, with a focus on the challenge of validating algorithmic output.

Text algorithms in economics
E Ash, S Hansen - Annual Review of Economics, 2023 - annualreviews.org
… methods used for algorithmic text analysis in economics, with … encompass most text-as-data research in economics and … with a focus on the challenge of validating algorithmic output. …
☆ 保存 99 引用 被引用次数：41 相关文章 所有 6 个版本 »

- **Four problems**: (I) Measuring document similarity; (II) Concept detection; (III) How concepts are related; (IV) Associating text with metadata.
- **Two challenges**: (I) Validation; (II) Interpretability.
- **One future**: Large Language Models.



18

18

Application: Corporate Culture Measurement

Measuring Corporate Culture Using Machine Learning

Kai Li
Sauder School of Business, University of British Columbia

Feng Mai
School of Business, Stevens Institute of Technology

Rui Shen
Shenzhen Finance Institute, School of Management and Economics
The Chinese University of Hong Kong, Shenzhen

Xinyan Yan
School of Business Administration, University of Dayton

We create a culture dictionary using one of the latest machine learning techniques—the word embedding model—and 209,480 earnings call transcripts. We score the five corporate cultural values of *innovation, integrity, quality, respect*, and *teamwork* for 62,664 firm-year observations over the period 2001–2018. We show that an innovative culture is broader than the usual measures of corporate innovation – R&D expenses and the number of patents. Moreover, we show that corporate culture correlates with business outcomes, including operational efficiency, risk-taking, earnings management, executive compensation design, firm value, and deal making, and that the culture-performance link is more pronounced in bad times. Finally, we present suggestive evidence that corporate culture is shaped by major corporate events, such as mergers and acquisitions. (*JEL* C45, G34, M14)

Received February 15, 2019; editorial decision May 26, 2020 by Editor Itay Goldstein. Authors have furnished an Internet Appendix, which is available on the Oxford University Press Web site next to the link to the final published paper online.

- Train **word embeddings** using word2vec on **earnings call transcripts**.
- Create a **culture dictionary** based on the **associations** of different words/phrases and the "**value words**" (*innovation, integrity, quality, respect, and teamwork*) under the trained word embeddings.
- **Business implications**: A strong corporate culture **correlates** with greater operational efficiency, more risk-taking, less earnings management, and higher firm value.

Measuring corporate culture using machine learning
K Li, F Mai, R Shen, X Yan
The Review of Financial Studies, 2021 · academic.oup.com

Abstract
We create a culture dictionary using one of the latest machine learning techniques—the word embedding model—and 209,480 earnings call transcripts. We score the five corporate cultural values of *innovation, integrity, quality, respect*, and *teamwork* for 62,664 firm-year observations over the period 2001–2018. We show that an innovative culture is broader than the usual measures of corporate innovation – R&D expenses and the number of patents. Moreover, we show that corporate culture correlates with business

展开

☆ 保存 引用 被引用次数：412 相关文章 所有 14 个版本 Web of Science: 119

19

19

Application: Product2Vec

Product2Vec: Leveraging representation learning to model consumer product choice in large assortments

NYU Stern School of Business

83 Pages • Posted: 7 Feb 2020 • Last revised: 3 Jul 2022

Fanglin Chen
New York University (NYU) - New York University (NYU), Leonard N. Stern School of Business, Department of Marketing, Students

Xiao Liu
New York University (NYU) - Leonard N. Stern School of Business

Davide Proserpio
Marshall School of Business - University of Southern California

Isamar Troncoso
Harvard Business School

Date Written: July 1, 2022

Abstract

We propose a method, Product2Vec, based on representation learning, that can automatically learn latent product attributes that drive consumer choices, to study product-level competition when the number of products is large. We demonstrate Product2Vec's interpretability and capability for scalable causal inference. For interpretability, first, we theoretically demonstrate that there exists a direct link between product vectors and product attributes by deriving a formal proof. Second, we use product embedding to create two metrics, complementarity and exchangeability, that allow us to distinguish between products that are complements and substitutes, respectively. For causal inference, we combine product vectors with choice models and show that we can achieve better accuracy—both in terms of model fit and unbiased price coefficients—when compared to a model based solely on observable attributes, and obtain results similar to those obtained with a more complex model that includes a fixed effect for every product.

Keywords: machine learning, product competition, representation learning, choice models

- Leverage the idea of skip-gram to learn the **latent product embeddings** based on other products in the **choice set**.
- **Interpretability**: A **theoretical proof** on the link between product embeddings, product attributes, and product clusters; **metrics** of complementarity and exchangeability.
- **Causal inference**: Combining product embeddings and choice models achieves **better accuracy** in **model fit** and **unbiased price coefficients estimation**.

Product2Vec: Leveraging representation learning to model consumer product choice in large assortments
F Chen, X Liu, D Proserpio… - NYU Stern School of …, 2022 - papers.ssrn.com
We propose a method, **Product2Vec**, based on representation learning, that can automatically learn latent product attributes that drive consumer choices, to study product-level …

☆ Save 引用 Cite Related articles

20

20

Application: Product2Vec-Email

RESEARCH-ARTICLE

E-commerce in Your Inbox: Product Recommendations at Scale

Authors: Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, Doug Sharp Authors Info & Claims

KDD '15: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining • August 2015 • Pages 1809–1818 • https://doi.org/10.1145/2783258.2788627

Published: 10 August 2015 Publication History

166 2,043

eReader PDF

ABSTRACT

In recent years online advertising has become increasingly ubiquitous and effective. Advertisements shown to visitors fund sites and apps that publish digital content, manage social networks, and operate e-mail services. Given such large variety of internet resources, determining an appropriate type of advertising for a given platform has become critical to financial success. Native advertisements, namely ads that are similar in look and feel to content, have had great success in news and social feeds. However, to date there has not been a winning formula for ads in e-mail clients. In this paper we describe a system that leverages user purchase history determined from e-mail receipts to deliver highly personalized product ads to Yahoo Mail users. We propose to use a novel neural language-based algorithm specifically tailored for delivering effective product recommendations, which was evaluated against baselines that included showing popular products and products predicted based on co-occurrence. We conducted rigorous offline testing using a large-scale product purchase data set, covering purchases of more than 29 million users from 172 e-commerce websites. Ads in the form of product recommendations were successfully tested on online traffic, where we observed a steady 9% lift in click-through rates over other ad formats in mail, as well as comparable lift in conversion rates. Following successful tests, the system was launched into production during the holiday season of 2014.

- Leverage the idea of skip-gram to learn the latent product embeddings based on purchases from email promotions.
- Train both product2vec and user2vec.
- **Impact**: 9%+ lift in CTR; launched into production.

E-commerce in your inbox: Product recommendations at scale

M Grbovic, V Radosavljevic, N Djuric, N Bhamidipati, J Savla, V Bhagwan, D Sharp

Proceedings of the 21th ACM SIGKDD international conference on knowledge …, 2015 · dl.acm.org

In recent years online advertising has become increasingly ubiquitous and effective. Advertisements shown to visitors fund sites and apps that publish digital content, manage social networks, and operate e-mail services. Given such large variety of internet resources, determining an appropriate type of advertising for a given platform has become critical to financial success. Native advertisements, namely ads that are similar in look and feel to content, have had great success in news and social feeds. However, to date there

SHOW MORE

☆ Save 99 Cite Cited by 385 Related articles All 9 versions »

21

21

Other Applications of Product2Vec

P2V-MAP: Mapping market structures for large retail assortments

S Gabel, D Guhl, D Klapper

Journal of Marketing Research, 2019 · journals.sagepub.com

The authors propose a new, exploratory approach for analyzing market structures that leverages two recent methodological advances in natural language processing and machine learning. They customize a neural network language model to derive latent product attributes by analyzing the co-occurrences of products in shopping baskets. Applying dimensionality reduction to the latent attributes yields a two-dimensional product map. This method is well-suited to retailers because it relies on data that are readily

SHOW MORE

☆ Save 99 Cite Cited by 61 Related articles All 5 versions Web of Science: 24 »

Scalable bundling via dense product embeddings

M Kumar, D Eckles, S Aral

arXiv preprint arXiv:2002.00100, 2020 · arxiv.org

Bundling, the practice of jointly selling two or more products at a discount, is a widely used strategy in industry and a well examined concept in academia. Historically, the focus has been on theoretical studies in the context of monopolistic firms and assumed product relationships, e.g., complementarity in usage. We develop a new machine-learning-driven methodology for designing bundles in a large-scale, cross-category retail setting. We leverage historical purchases and consideration sets created from clickstream data to

展开

☆ 保存 99 引用 被引用次数：10 相关文章 所有 3 个版本 »

Item2vec: neural item embedding for collaborative filtering

O Barkan, N Koenigstein

2016 IEEE 26th International Workshop on Machine Learning for …, 2016 · ieeexplore.ieee.org

Many Collaborative Filtering (CF) algorithms are item-based in the sense that they analyze item-item relations in order to produce item similarities. Recently, several works in the field of Natural Language Processing (NLP) suggested to learn a latent representation of words using neural embedding algorithms. Among them, the Skip-gram with Negative Sampling (SGNS), also known as word2vec, was shown to provide state-of-the-art results on various linguistics tasks. In this paper, we show that item-based CF can be cast in the

SHOW MORE

☆ Save 99 Cite Cited by 600 Related articles All 13 versions »

22

22

Agenda

- Word2Vec: Continuous Bag of Words (CBOW), Skip-Gram, Evaluations, Applications
- **RNN, LSTM, Seq2Seq, and Applications**
- Attention Mechanism and Transformer

23

23

Why Does Sequence Matter?

- Consider the following case:

This is not a real restaurant, it's a filthy burger joint.

This is not a filthy burger joint, it's a real restaurant.

- If we only use word embeddings for sentiment classification, these two opposite sentences will generate the same sentence vector (i.e., the sentence vector is the **sum** of the word vectors for all words in the sentence).

24

24

Back to N-Gram Models

- N-gram model is a **language model** that limits the dependencies on history, a.k.a. **Markov Chain**.

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\underbrace{\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)}}_{n\text{-1 words}})$$ (assumption)

prob of a n-gram / prob of a (n-1)-gram

$$= \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$ (definition of conditional prob)

Two important issues:

- Sparsity (partially addressed by smoothing)
- Model Size (n is no more than 5, usually 2 or 3)

- **Question:** How do we get these *n*-gram and (*n*-1)-gram probabilities?
- **Answer:** By counting them in some large corpus of text!

$$\approx \frac{\text{count}(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$ (statistical approximation)

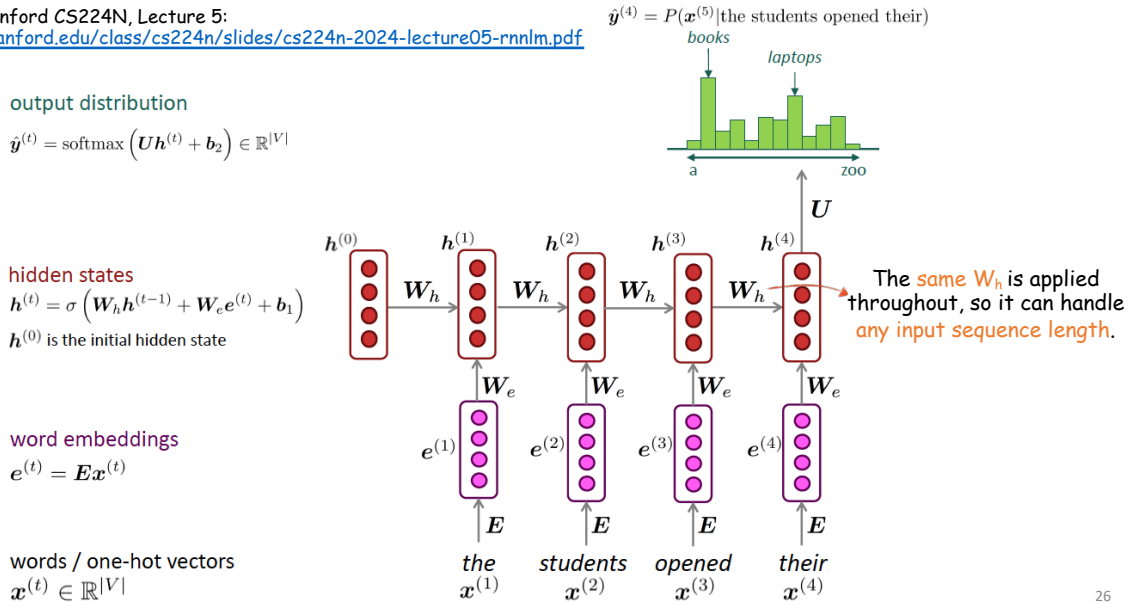
- N-gram models that "work" in the era of LLM: https://arxiv.org/abs/2401.17377

25

25

Recurrent Neural Network (RNN)

Reference: Stanford CS224N, Lecture 5: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf



26

26

Training RNN

Reference: Stanford CS224N, Lecture 5:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf

- Loss functions in step t is the cross-entropy between the true 1-hot and the predicted distribution:

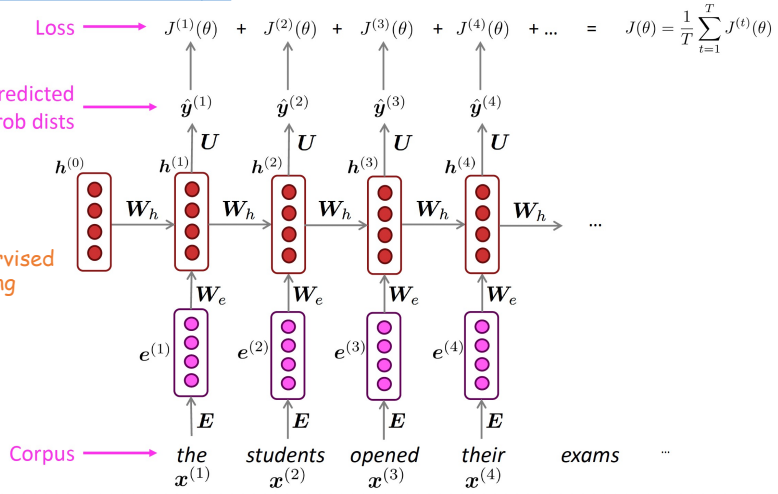
$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

- So, the overall loss for the entire training corpus is:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

Self-Supervised Learning

- In practice, we leverage the idea of SGD: Compute loss and gradients, and update weights with batches of words/sentences. Then repeat on a new batch of sentences.



27

27

Vanishing (and Exploding) Gradient in RNN

Reference: Stanford CS224N, Lecture 5:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf

Backpropagation through time

$$\frac{\partial L}{\partial W_h} \propto \sum_{1 \le k \le t} \left(\prod_{t \ge i > k} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W_h}$$

Contribution of hidden state *k*

Length of the product proportional to how far *k* is from *t*

$$\frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial h_{t-3}} \frac{\partial h_{t-3}}{\partial h_{t-4}} \frac{\partial h_{t-4}}{\partial h_{t-5}} \frac{\partial h_{t-5}}{\partial h_{t-6}} \frac{\partial h_{t-6}}{\partial h_{t-7}} \frac{\partial h_{t-7}}{\partial h_{t-8}} \frac{\partial h_{t-8}}{\partial h_{t-9}} \frac{\partial h_{t-9}}{\partial h_{t-10}} \frac{\partial h_{t-10}}{\partial W_h}$$

Contribution of hidden state *t-10*

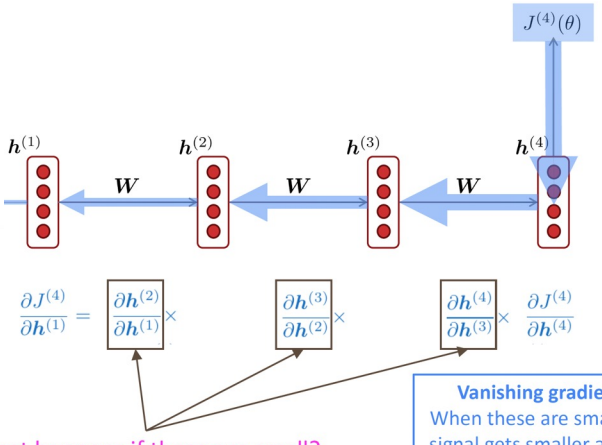
Vanishing vs. Exploding Gradient

On the difficulty of training recurrent neural networks
R Pascanu, T Mikolov, Y Bengio
International conference on machine learning, 2013 · proceedings.mlr.press

Abstract
There are two widely known issues with properly training recurrent neural networks, the vanishing and the exploding gradient problems detailed in Bengio et al.(1994). In this paper we attempt to improve the understanding of the underlying issues by exploring these problems from an analytical, a geometric and a dynamical systems perspective. Our analysis is used to justify a simple yet effective solution. We propose a gradient norm clipping strategy to deal with exploding gradients and a soft constraint for the vanishing

SHOW MORE

☆ Save 99 Cite Cited by 6901 Related articles All 11 versions »



What happens if these are small?

Gradient signals from far away will be lost!
The weights W_h only capture near effects.

Vanishing gradient problem:
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

28

28

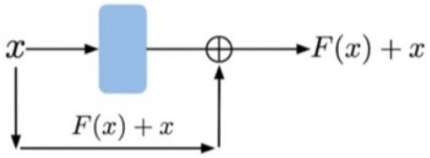
Gradient Clipping and Skip Connection

Reference: Stanford CS224N, Lecture 5:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture05-rnnlm.pdf

- The exploding gradient problem is relatively easier to address: Gradient Clipping.
- Intuition: Take a smaller step in the same direction.
- One idea to address vanishing gradient is to create direct and linear pass-through connections in the model: Residual/skip connections, attention, etc.

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if



Deep residual learning for image recognition
K He, X Zhang, S Ren, J Sun - … and pattern recognition, 2016 - openaccess.thecvf.com
… Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. …
★ 保存 99 引用 被引用次数：196717 相关文章 所有 76 个版本 »

29

29

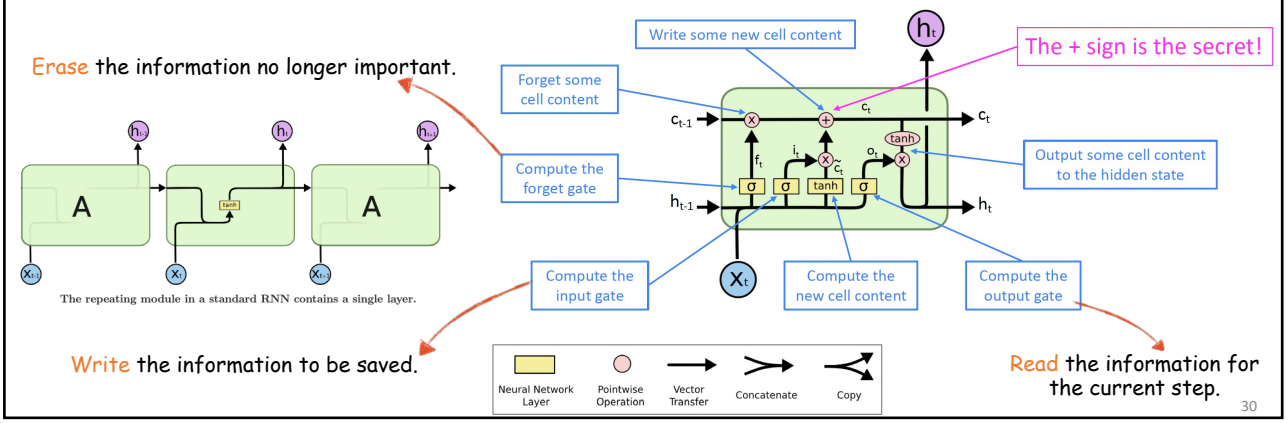
LSTM Model

Long short-term memory
S Hochreiter, J Schmidhuber - Neural computation, 1997 - ieeexplore.ieee.org
… (**short-term memory**, as opposed to **long**-term **memory** … learning what to put in **shortterm memory**, however, take too … and corresponding teacher signals are **long**. Although theoretically …
☆ Save 99 Cite Cited by 98734 Related articles All 45 versions »

Reference: Stanford CS224N, Lecture 6:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM was the dominant approach for most NLP tasks in 2013-2015.

- LSTM: Disregard the irrelevant past information based on the current information.
- LSTM has a hidden state $h^{(t)}$ and a cell state $c^{(t)}$, which stores long-term information.



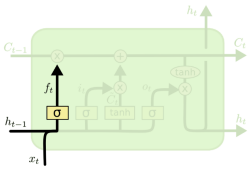
30

30

LSTM Model Details

Reference: Stanford CS224N, Lecture 6:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

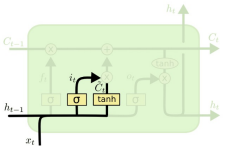
- Step 1: Decide how to forget the prior information.



Forget Gate

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

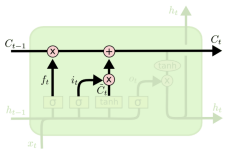
- Step 2: Decide the information needed to be stored to the cell.



Input Gate

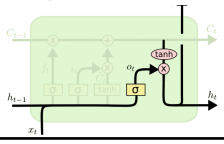
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

- Step 3: Update the cell information.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- Step 4: Decide the output.



Output Gate

$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

31

31

Evaluating Language Models

Reference: Stanford CS224N, Lecture 6:
https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf
https://engineering.fb.com/2016/10/25/ml-applications/building-an-efficient-neural-language-model-over-a-billion-words/

- Perplexity: The standard metric for evaluating a language model.

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

- Perplexity is the exponential of the cross-entropy loss, so the lower the better:

$$\text{Perplexity} = \prod_{t=1}^{T} \left(\frac{1}{\hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}} \right) = \exp(J(\theta))$$

RNNs greatly improve perplexity over n-grams.

5-Gram

DNN-based Methods
(RNN, LSTM, etc.)

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

Table 2. Comparison on 1B word in perplexity (lower the better). Note that Jozefowicz et al., uses 32 GPUs for training. We only use 1 GPU.

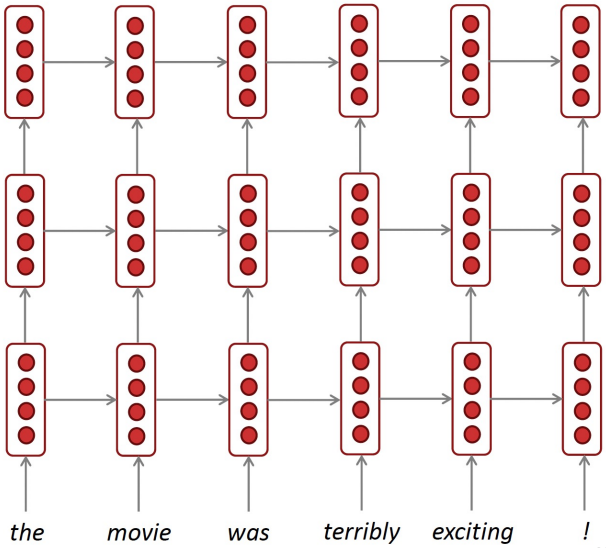
32

32

Multi-layer (Stacked) RNN

Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf

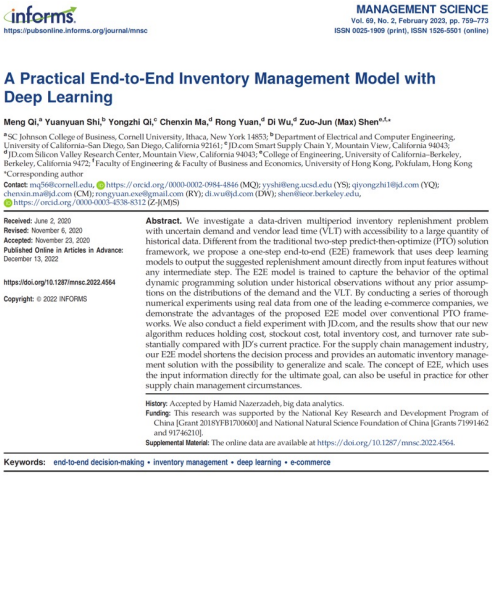
- Multi-layer RNN: Deep beyond the time dimension.
- In neural machine translation, 2-4 layers is best for encoder RNNs; 4 layers is best for decoder RNNs.
- 2 layers is much better than 1, but 3 may be a little better than 2.
- Transformer-based nets are usually much deeper.



33

33

Application of RNN: Inventory Management



informs
https://pubsonline.informs.org/journal/mnsc

MANAGEMENT SCIENCE
Vol. 69, No. 2, February 2023, pp. 759–773
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

A Practical End-to-End Inventory Management Model with Deep Learning

Meng Qi,[a] Yuanyuan Shi,[b] Yongzhi Qi,[c] Chenxin Ma,[d] Rong Yuan,[d] Di Wu,[d] Zuo-Jun (Max) Shen[e,*]

[a] SC Johnson College of Business, Cornell University, Ithaca, New York 14853; [b] Department of Electrical and Computer Engineering, University of California–San Diego, San Diego, California 92161; [c] JD.com Smart Supply Chain Y, Mountain View, California 94043; [d] JD.com Silicon Valley Research Center, Mountain View, California 94043; [e] College of Engineering, University of California–Berkeley, Berkeley, California 9472; [f] Faculty of Engineering & Faculty of Business and Economics, University of Hong Kong, Pokfulam, Hong Kong
*Corresponding author

Contact: mq56@cornell.edu, https://orcid.org/0000-0002-0984-4846 (MQ); yyshi@eng.ucsd.edu (YS); qiyongzhi1@jd.com (YQ); chenxin.ma@jd.com (CM); rongyuan.exe@gmail.com (RY); di.wu@jd.com (DW); shen@ieor.berkeley.edu, https://orcid.org/0000-0003-4538-8312 (Z-JMS)

Received: June 2, 2020
Revised: November 6, 2020
Accepted: November 23, 2020
Published Online in Articles in Advance: December 13, 2022

https://doi.org/10.1287/mnsc.2022.4564

Copyright: © 2022 INFORMS

Abstract. We investigate a data-driven multiperiod inventory replenishment problem with uncertain demand and vendor lead time (VLT) with accessibility to a large quantity of historical data. Different from the traditional two-step predict-then-optimize (PTO) solution framework, we propose a one-step end-to-end (E2E) framework that uses deep learning models to output the suggested replenishment amount directly from input features without any intermediate step. The E2E model is trained to capture the behavior of the optimal dynamic programming solution under historical observations without any prior assumptions on the distributions of the demand and the VLT. By conducting a series of thorough numerical experiments using real data from one of the leading e-commerce companies, we demonstrate the advantages of the proposed E2E model over conventional PTO frameworks. We also conduct a field experiment with JD.com, and the results show that our new algorithm reduces holding cost, stockout cost, total inventory cost, and turnover rate substantially compared with JD's current practice. For the supply chain management industry, our E2E model shortens the decision process and provides an automatic inventory management solution with the possibility to generalize and scale. The concept of E2E, which uses the input information directly for the ultimate goal, can also be useful in practice for other supply chain management circumstances.

History: Accepted by Hamid Nazerzadeh, big data analytics.
Funding: This research was supported by the National Key Research and Development Program of China [Grant 2018YFB1700600] and National Natural Science Foundation of China [Grants 71991462 and 91746210].
Supplemental Material: The online data are available at https://doi.org/10.1287/mnsc.2022.4564.

Keywords: end-to-end decision-making • inventory management • deep learning • e-commerce

- RNN is not that frequently used in business research, because it does NOT directly produce text representations.
- Use multi-quantile RNNs to provide end-to-end predictions from features to the optimal inventory decisions, whereas most of the literature applies the predict-then-optimize paradigm.
- A field experiment shows that the e2e approach substantially reduces the inventory costs compared with some naïve benchmarks.

A practical end-to-end inventory management model with deep learning
M Qi, Y Shi, Y Qi, C Ma, R Yuan, D Wu, ZJ Shen
Management Science, 2023 · pubsonline.informs.org

We investigate a data-driven multiperiod inventory replenishment problem with uncertain demand and vendor lead time (VLT) with accessibility to a large quantity of historical data. Different from the traditional two-step predict-then-optimize (PTO) solution framework, we propose a one-step end-to-end (E2E) framework that uses deep learning models to output the suggested replenishment amount directly from input features without any intermediate step. The E2E model is trained to capture the behavior of the optimal dynamic

SHOW MORE

☆ Save 99 Cite Cited by 65 Related articles All 4 versions Web of Science: 5 »

34

34

Application of RNN: Detecting FTD

Psychiatry Research 304 (2021) 114135

Contents lists available at ScienceDirect

Psychiatry Research

journal homepage: www.elsevier.com/locate/psychres

Detecting formal thought disorder by deep contextualized word representations

Justyna Sarzynska-Wawer [*,a], Aleksander Wawer [1,b], Aleksandra Pawlak [2,c], Julia Szymanowska [2,c], Izabela Stefaniak [d], Michal Jarkiewicz [3,d], Lukasz Okruszek [a]

[a] Institute of Psychology, Polish Academy of Sciences, Jaracza 1, 00-378 Warszawa, Poland
[b] Institute of Computer Science, Polish Academy of Sciences, Jana Kazimierza 5, 01-248 Warszawa, Poland
[c] University of Social Sciences and Humanities, Chodakowska 19/31, 03-815 Warszawa, Poland
[d] Institute of Psychiatry and Neurology, Sobieskiego 9, 02-957 Warszawa, Poland

ARTICLE INFO

Keywords:
Schizophrenia
Language
Natural language processing
Deep learning

ABSTRACT

Computational linguistics has enabled the introduction of objective tools that measure some of the symptoms of schizophrenia, including the coherence of speech associated with formal thought disorder (FTD). Our goal was to investigate whether neural network based utterance embeddings are more accurate in detecting FTD than models based on individual indicators. The present research used a comprehensive Embeddings from Language Models (ELMo) approach to represent interviews with patients suffering from schizophrenia (N=35) and with healthy people (N=35). We compared its results to the approach described by Bedi et al. (2015), referred to here as the coherence model. Evaluations were also performed by a clinician using the Scale for the Assessment of Thought, Language and Communication (TLC). Using all six TLC questions the ELMo obtained an accuracy of 80% in distinguishing patients from healthy people. Previously used coherence models were less accurate at 70%. The classifying clinician was accurate 74% of the time. Our analysis shows that both ELMo and TLC are sensitive to the symptoms of disorganization in patients. In this study methods using text representations from language models were more accurate than those based solely on the assessment of FTD, and can be used as measures of disordered language that complement human clinical ratings.

- How to detect formal thought disorder (FTD)?
- Embeddings from LSTM language models (ELMo) can more accurately detect/predict FTD than individual indicators (benchmark: coherence model, which can somehow be viewed as traditional NLP method).
- Accuracy (N=70, 35 healthy and 35 patients):
  - ELMo: 80%
  - Coherence models: <70%
  - Clinician: 74%

[HTML] Detecting formal thought disorder by deep contextualized word representations

J Sarzynska-Wawer, A Wawer, A Pawlak… - Psychiatry …, 2021 - Elsevier

Computational linguistics has enabled the introduction of objective tools that measure some of the symptoms of schizophrenia, including the coherence of speech associated with formal thought disorder (FTD). Our goal was to investigate whether neural network based utterance embeddings are more accurate in detecting FTD than models based on individual indicators. The present research used a comprehensive Embeddings from Language Models (ELMo) approach to represent interviews with patients suffering from schizophrenia …

☆ Save 99 Cite Cited by 14561 Related articles All 24 versions Web of Science: 74 »

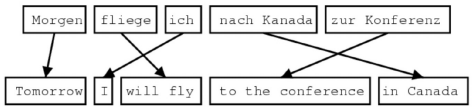
35

35

Neural Machine Translation (NMT)

Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf

- NMT is a way to do machine translation with a single end-to-end neural network: Sequence-to-sequence (seq2seq), which involves 2 RNNs.
- Machine translation is highly nontrivial and once was a huge research field in CS and NLP.



1519年600名西班牙人在墨西哥登陆，去征服几百万人口的阿兹特克帝国，初次交锋他们损兵三分之二。

In 1519, six hundred Spaniards landed in Mexico to conquer the Aztec Empire with a population of a few million. They lost two thirds of their soldiers in the first clash.

translate.google.com (2009): 1519 600 Spaniards landed in Mexico, millions of people to conquer the Aztec empire, the first two-thirds of soldiers against their loss.

translate.google.com (2013): 1519 600 Spaniards landed in Mexico to conquer the Aztec empire, hundreds of millions of people, the initial confrontation loss of soldiers two-thirds.

translate.google.com (2015): 1519 600 Spaniards landed in Mexico, millions of people to conquer the Aztec empire, the first two-thirds of the loss of soldiers they clash.

36

36

Seq2Seq for NMT

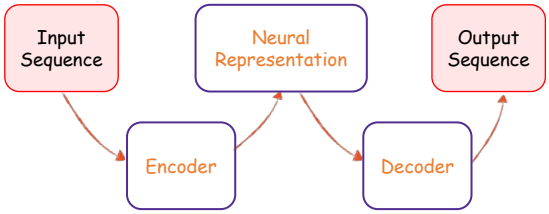
Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf

- Seq2seq is a Conditional Language Model:
  - Predicting the next word of the target sentence y conditioned on the source sentence x and prior texts.

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots\underbrace{P(y_T|y_1,\ldots,y_{T-1},x)}$$

Probability of next target word, given target words so far and source sentence *x*

- Encoder-decoder architecture:
  - Seq2seq: both input and output are sequences.
  - Summarization: Long text → short text
  - Dialogue: previous utterances → next utterance
  - Parsing: Input text → output parse as a sequence
  - Code generation → Natural language → Python code

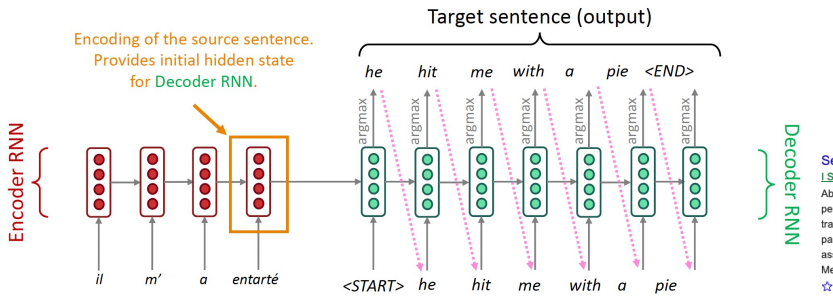


37

37

Seq2Seq Architecture

Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf



Sequence to sequence learning with neural networks
I Sutskever, O Vinyals, QV Le - Advances in neural ..., 2014 - proceedings.neurips.cc
Abstract Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then ...
☆ Save 99 Cite Cited by 25043 Related articles All 28 versions »

Encoder RNN produces an encoding of the source sentence.

Decoder RNN is a Language Model that generates target sentence, *conditioned on encoding*.

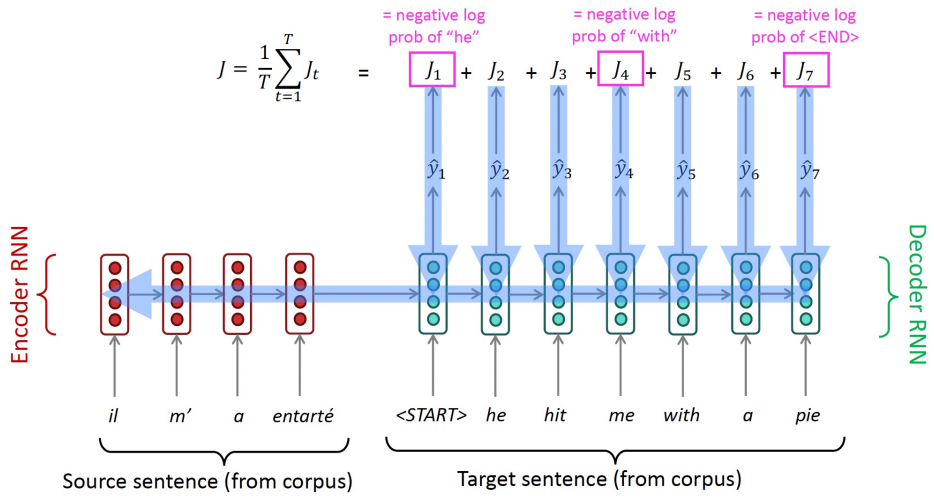
Note: This diagram shows **test time** behavior: decoder output is fed in ·····→ as next step's input

38

38

Seq2Seq Training

Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf



Seq2seq is optimized as a **single system.** Backpropagation operates "*end-to-end*".

39

39

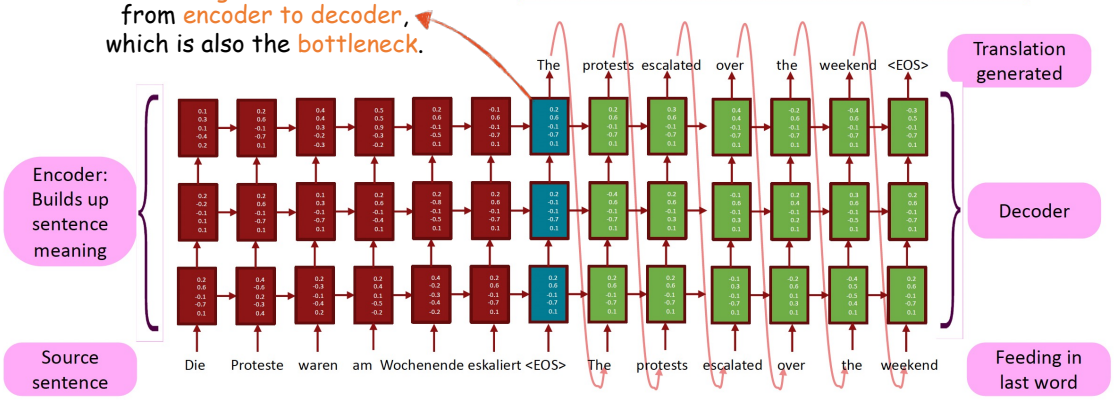
Multi-Layer Seq2Seq

Reference: Stanford CS224N, Lecture 6: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture06-fancy-rnn.pdf

[Sutskever et al. 2014; Luong et al. 2015]

The hidden states from RNN layer *i* are the inputs to RNN layer *i*+1

Conditioning: Information flow from encoder to decoder, which is also the bottleneck.



What's next: Attention Mechanism and Transformer!

40

40

NMT: The First Major Success of DL-NLP

Reference: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf

- NMT transformed from a mere research attempt in 2014 to the leading standard in 2016.
- 2014: The Seq2seq paper.
- 2016: Adopted by Google Translate.
- 2018: Adopted by everyone.

- The original statistical machine translation (SMT) system, built by hundreds of engineers over many years, soon outperformed by NMT trained by small groups of engineers in a few months.

41

41

Agenda

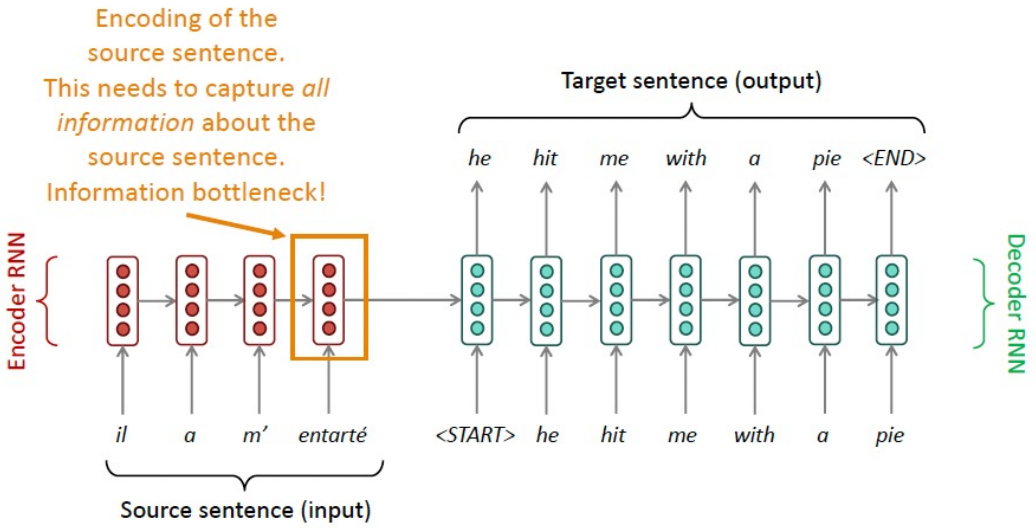
- Word2Vec: Continuous Bag of Words (CBOW), Skip-Gram, Evaluations, Applications
- RNN, LSTM, Seq2Seq, and Applications
- **Attention Mechanism and Transformer**

42

42

Information Bottleneck in RNN

Reference: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf



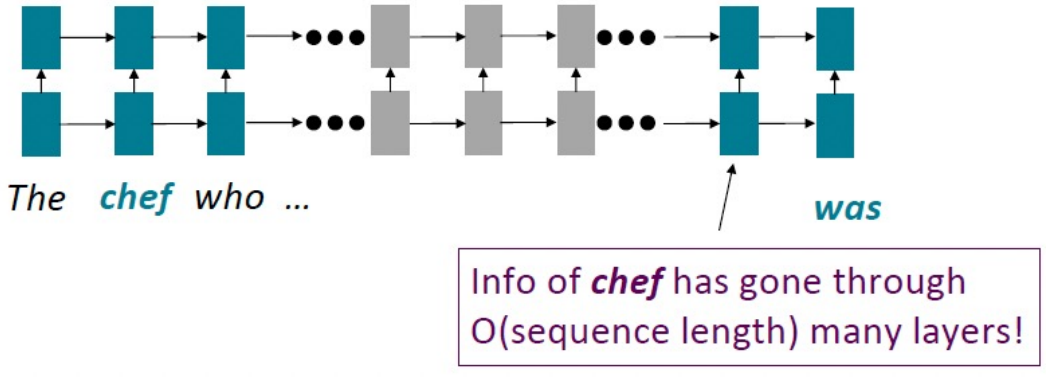
43

43

Issue with RNN: Linear Interaction Distance

Reference: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf

- Human languages are intrinsically NOT linearly ordered.



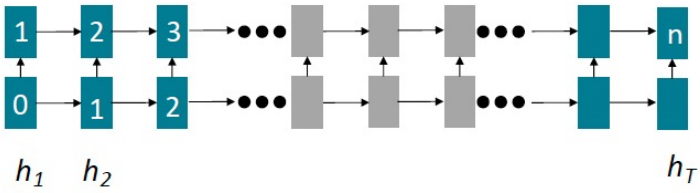
44

44

Issue with RNN: Non-parallelizability

Reference: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf

- Forward and backward passes both have O(sequence length) unparallelizable operations.
- GPUs can perform independent small computations quickly in a large scale.
- Future hidden states cannot be computed (in full) before past RNN hidden states have been computed.
- Cannot scale with a very large dataset.



Numbers indicate min # of steps before a state can be computed

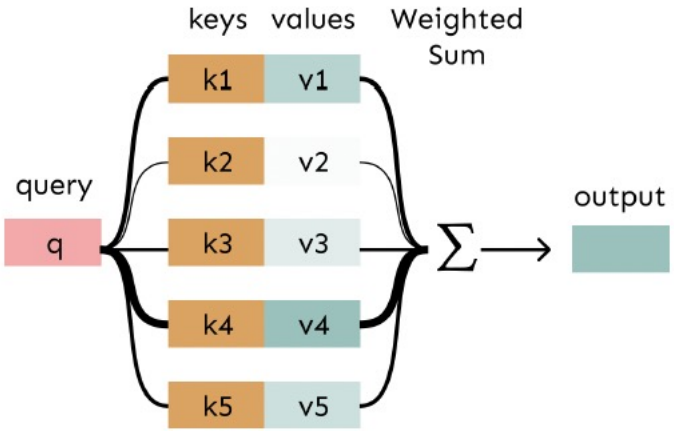
45

45

Attention as a Very General DL Technique

Reference: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf

- **Attention**: A weighted sum of the values dependent on the query.
  - A selective summary of the information contained in the values, where the query determines which values to focus on.



46

46

Attention is All You Need

References: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf
https://peterbloem.nl/blog/transformers

- Transformer: No RNN architecture, just attention mechanism.
- Self-attention: To generate y_t, we need to pay attention to $y_{<t}$.

$$w'_{ij} = \frac{\mathbf{q}_i^T \mathbf{k}_j}{\sqrt{k}}$$

Query $\mathbf{q}_i = \mathbf{W}_q \mathbf{x}_i$ Key $\mathbf{k}_i = \mathbf{W}_k \mathbf{x}_i$ Value $\mathbf{v}_i = \mathbf{W}_v \mathbf{x}_i$

$$w'_{ij} = \mathbf{q}_i^T \mathbf{k}_j$$

$$w_{ij} = \text{softmax}(w'_{ij})$$

$$\mathbf{y}_i = \sum_j w_{ij} \mathbf{v}_j \ .$$

Why does it work?

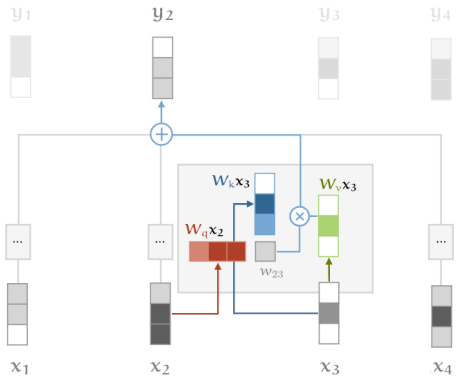


Illustration of the self-attention with key, query and value transformations.

Attention is all you need
A Vaswani, N Shazeer, N Parmar... - Advances in neural ..., 2017 - proceedings.neurips.cc
... to attend to all positions in the decoder up to and including that position. We need to prevent ... We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) ...
☆ 保存 99 引用 被引用次数：109517 相关文章 所有 62 个版本 »

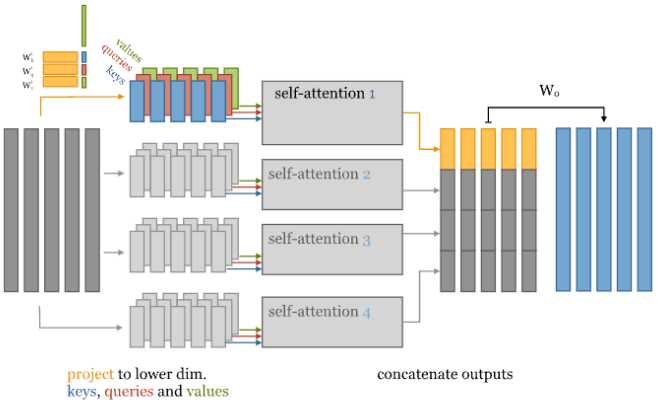
47

47

Multi-head Attention

References: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf
https://peterbloem.nl/blog/transformers

- Multi-head attention is a way to speed up the training procedure.
- Instead of using a large matrix to compute all attentions, we can compute multiple attention matrices and concatenate the final vectors.
- Allows for parallel computing: Deploy attention mechanisms to multiple computing cores in parallel and sum them up at the end.
- Input dim = 256, 8 attention heads, each with 32 dimensions.



The basic idea of multi-head self-attention with 4 heads. To get our keys, queries and values, we project the input down to vector sequences of smaller dimension.

48

48

Position Encoding

References: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf
https://peterbloem.nl/blog/transformers

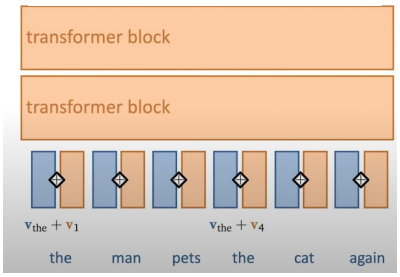
word embeddings:

$\mathbf{v}_{\text{the}}, \mathbf{v}_{\text{man}}, \mathbf{v}_{\text{pets}}, \mathbf{v}_{\text{cat}}, \mathbf{v}_{\text{again}}$

- Position embeddings: Position vectors which are learned.
- Position encoding: The function from position to vector.
- The final input of the model is the sum of word embeddings and position embeddings.

position embeddings:

$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4, \mathbf{v}_5, \ldots$



49

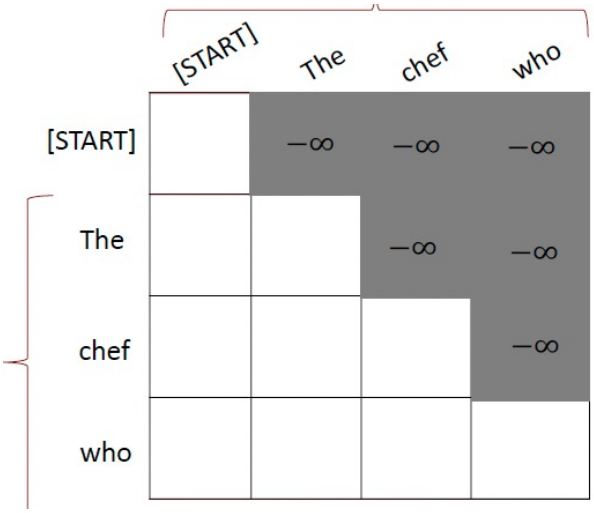
49

Auto-Regression

References: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf
https://peterbloem.nl/blog/transformers

- Self-supervised learning for transformers.
- To use self-attention in decoders, we need to mask the future.
- Inefficient implementation: Change the set of keys and queries to include only past words.
- Parallelizable implementation: Mask out attention to future words by setting the weight to -inf.

$$w'_{ij} = \begin{cases} q_i^T k_j, j \leq i \\ -\infty, j > i \end{cases}$$



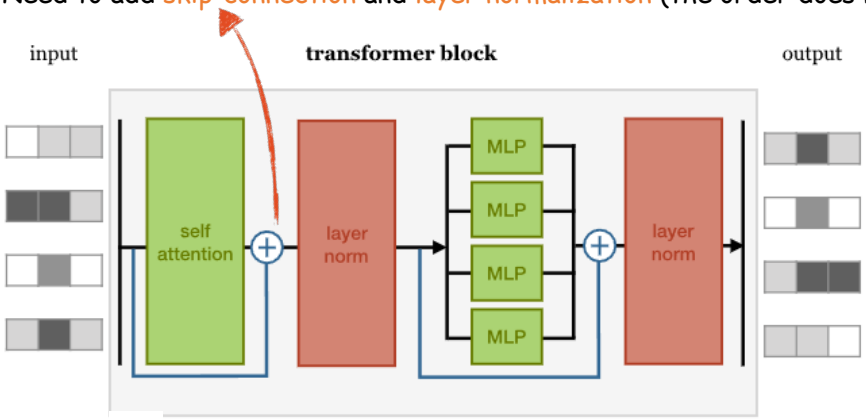
50

50

Transformer

References: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf
https://peterbloem.nl/blog/transformers

- Transformer = Multi-head self-attention + MLP + position encoding + autoregression
- Need to add skip-connection and layer normalization (the order does not matter).



51

51

Layer Normalization

Layer normalization
JL Ba, JR Kiros, GE Hinton - arXiv preprint arXiv:1607.06450, 2016 - arxiv.org
… , we transpose batch **normalization** into **layer normalization** by computing the mean and variance used for **normalization** from all of the summed inputs to the neurons in a **layer** on a …
☆ Save 99 Cite Cited by 10350 Related articles All 6 versions »

References: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf
https://peterbloem.nl/blog/transformers

- Layer normalization: A trick to help models train faster.
- Cut down on uninformative variation in hidden values by normalizing to unit mean and standard deviation within each layer → Normalized gradients.
  - Let $x \in \mathbb{R}^d$ be an individual (word) vector in the model.
  - Let $\mu = \sum_{j=1}^{d} x_j$; this is the mean; $\mu \in \mathbb{R}$.
  - Let $\sigma = \sqrt{\frac{1}{d}\sum_{j=1}^{d}(x_j - \mu)^2}$; this is the standard deviation; $\sigma \in \mathbb{R}$.
  - Let $\gamma \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$ be learned "gain" and "bias" parameters. (Can omit!)
  - Then layer normalization computes:

$$\text{output} = \frac{x - \mu}{\sqrt{\sigma + \epsilon}} * \gamma + \beta$$

Normalize by scalar mean and variance

Modulate by learned elementwise gain and bias

52

52

Attention is All You Need

References: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf
https://peterbloem.nl/blog/transformers

- Input: Sequence in language one and Sequence in language two.
- Architecture: Encoder + Decoder
- 8 heads, 512 embedding dimensions, 2048 sentence length
- Trained on 8 GPUs for 5 days.

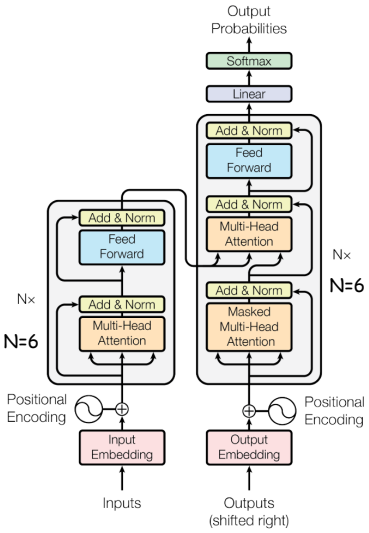


Figure 1: The Transformer - model architecture.

53

53

Attention is All You Need

References: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf
https://peterbloem.nl/blog/transformers

Machine Translation

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

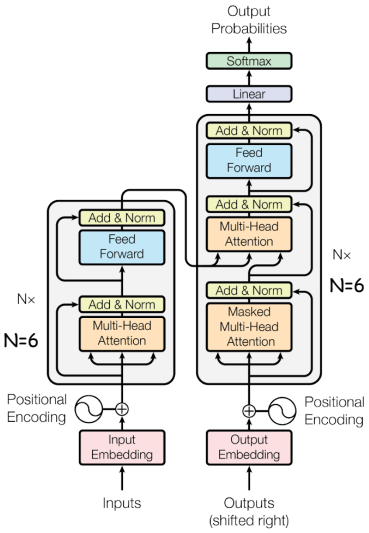


Figure 1: The Transformer - model architecture.

54

54

Attention is All You Need

References: Stanford CS224N, Lecture 8: https://web.stanford.edu/class/cs224n/slides/cs224n-2024-lecture08-transformers.pdf
https://peterbloem.nl/blog/transformers

Document Generation

Model	Test perplexity	ROUGE-L
seq2seq-attention, L = 500	5.04952	12.7
Transformer-ED, L = 500	2.46645	34.2
Transformer-D, L = 4000	2.22216	33.6
Transformer-DMCA, no MoE-layer, L = 11000	2.05159	36.2
Transformer-DMCA, MoE-128, L = 11000	1.92871	37.9
Transformer-DMCA, MoE-256, L = 7500	1.90325	38.8

The old standard

Transformers all the way down.

The parallelizability of transformer enables large-scale pre-training!

Figure 1: The Transformer - model architecture.

55

55