Language Models

VU VRIJE UNIVERSITEIT AMSTERDAM

Language model

- A language model is a probabilistic model for guessing the probabilities of words in a sequence
- Initially, models were designed to do only one task (e.g., word2vec)
- Most recent models generalize to different tasks and languages (e.g., BERT)
- **EXTREME benchmark**: Test considering 40 languages and 9 taskss

LINK

word2vec

- Proposed by Google in **2013**. Highly influential!
- These vectors are constructed following the principle*: "You shall know a word by the company it keeps"* Firth, 1957

- word2vec maps *words* in a high-dimensional continuous vector space
  - Each word is mapped to a vector of real number called **embedding**

- The vectors can be constructed using two models:
  - **Skip-gram**
  - **Continuous bag-of-words (CBOW)**

word2vec

- The vector embeddings can be used for a variety of reasons:
  - Word similarity
  - Analogies
  - …
- Example: Vector(France) + Vector(Paris) – Vector(Italy) = ?

DEMO

word2vec

Skip-gram model

- *Goal:* Given one word, predict the ones around it

- We can look at words that appear around *w* in an existing corpus and train a model that, given *w*, returns the best scores for words such as the ones we observed in our corpus

word2vec

Skip-gram model

- *Goal:* Given one word, predict the ones around it

- More formally, we want to maximize a function like

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

word2vec

Skip-gram model

- How can we implement the conditional probability *p*?
- We assign two vectors of real numbers ${v'}_{W_I}$ and v_{W_O} for each word in the vocabulary *W* and calculate *p* as:

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

Machine Learning digression

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

- Remember: Our goal is to maximize

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t)$$

- How can we learn good values of v'_{W_I} and v_{W_O} to this end?
- Why is *p* so complex?

VU VRIJE UNIVERSITEIT AMSTERDAM

Machine Learning digression

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

- Remember: Our goal is to maximize

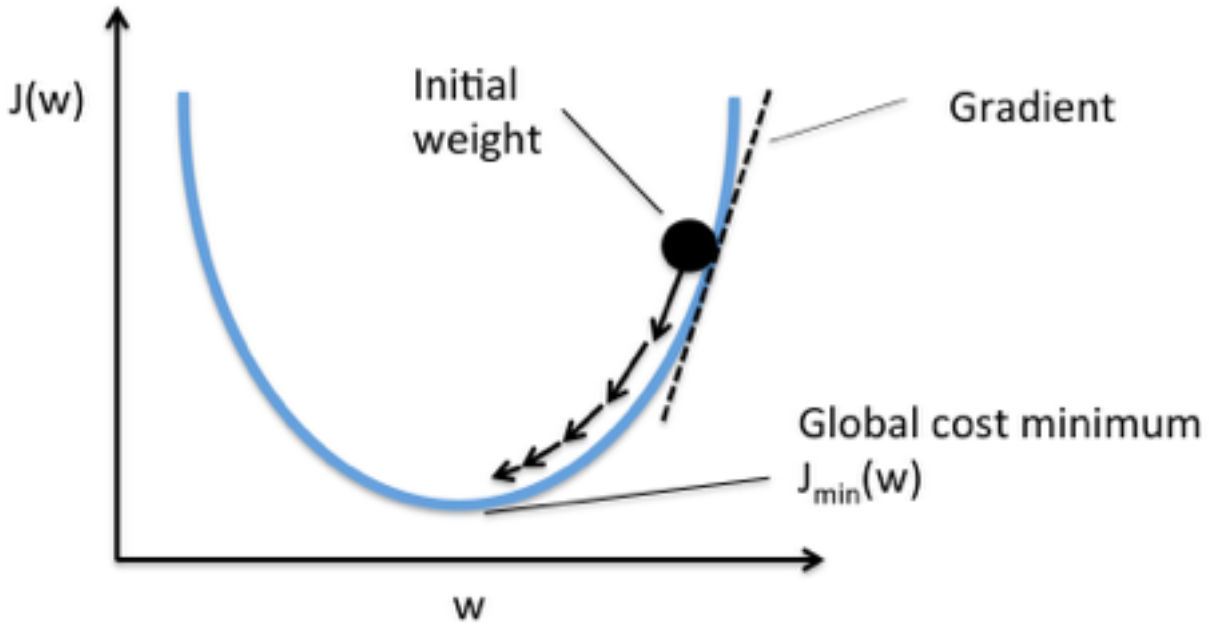
$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t)$$

- **How can we learn good values of v'_{W_I} and v_{W_O} to this end?**
- Why is *p* so complex?

Machine Learning digression

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

- Remember: The **gradient** of a function tells a direction of change



VU VRIJE UNIVERSITEIT AMSTERDAM

Machine Learning digression

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

- Remember: Our goal is to maximize

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t)$$

- How can we learn good values of v'_{W_I} and v_{W_O} to this end?
- **Why is *p* so complex?**

VU VRIJE UNIVERSITEIT AMSTERDAM

Machine Learning digression

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

- In machine learning, such function is called *softmax*
- Even though, at first sight it looks unnecessary complex, in practice it has very useful properties:
  - Sum is 1 (probabilities)
  - Calculating the gradient is very simple: $\frac{d(e^x)}{dx} = e^x$

VU VRIJE UNIVERSITEIT AMSTERDAM

word2vec

Skip-gram model

- The original formulation of

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$

  is expensive to calculate because it is proportional to the size of the dictionary (W)

- *Two* techniques improve the execution: *Hierarchical softmax* and *Negative Sampling*

word2vec

Continuous Bag-of-Words (CBOW)

- Alternative method to perform the same task

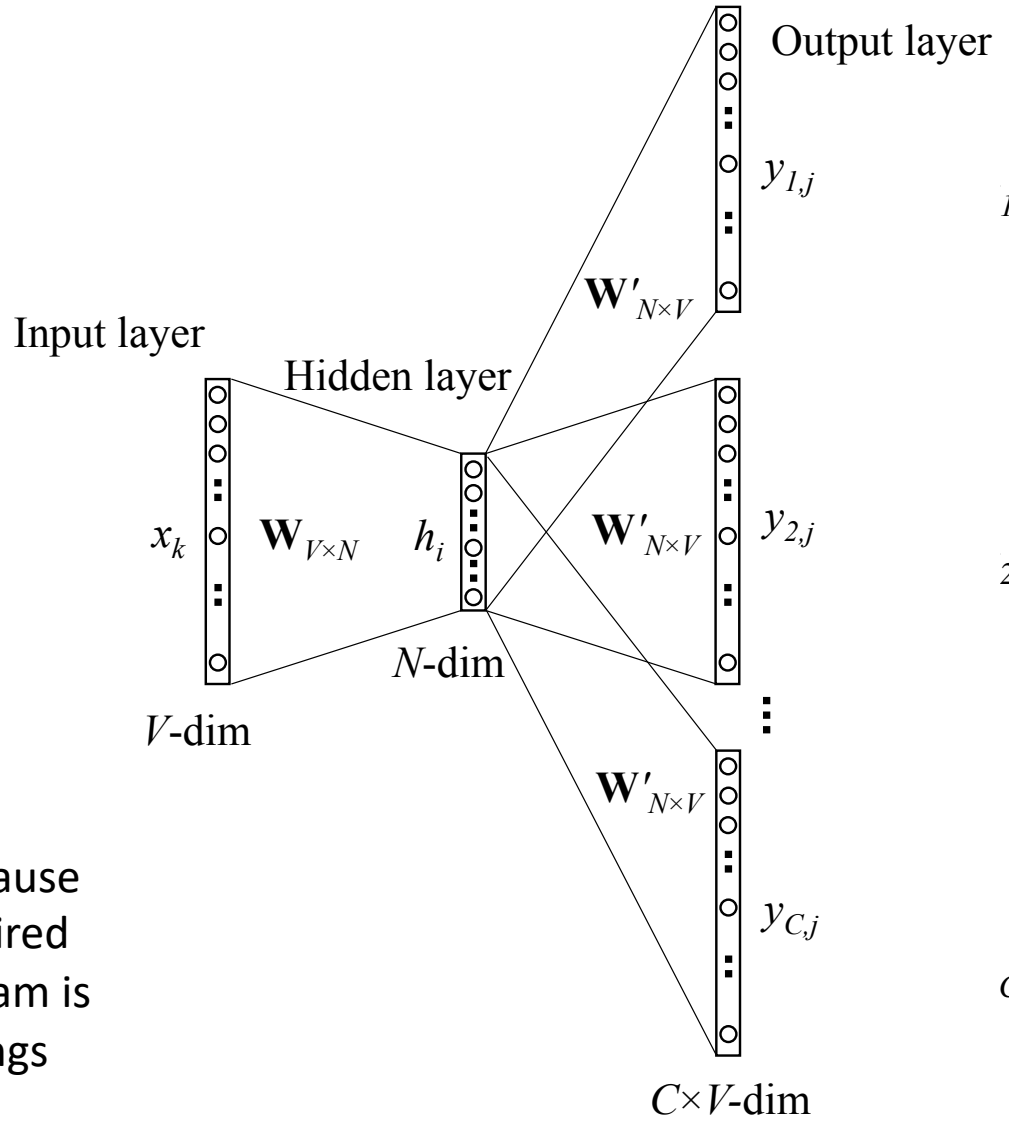
- In this case, the prediction is different: Given a context (w1, w2, w3,w4, w5), what's the probability that a new word appear?

word2vec

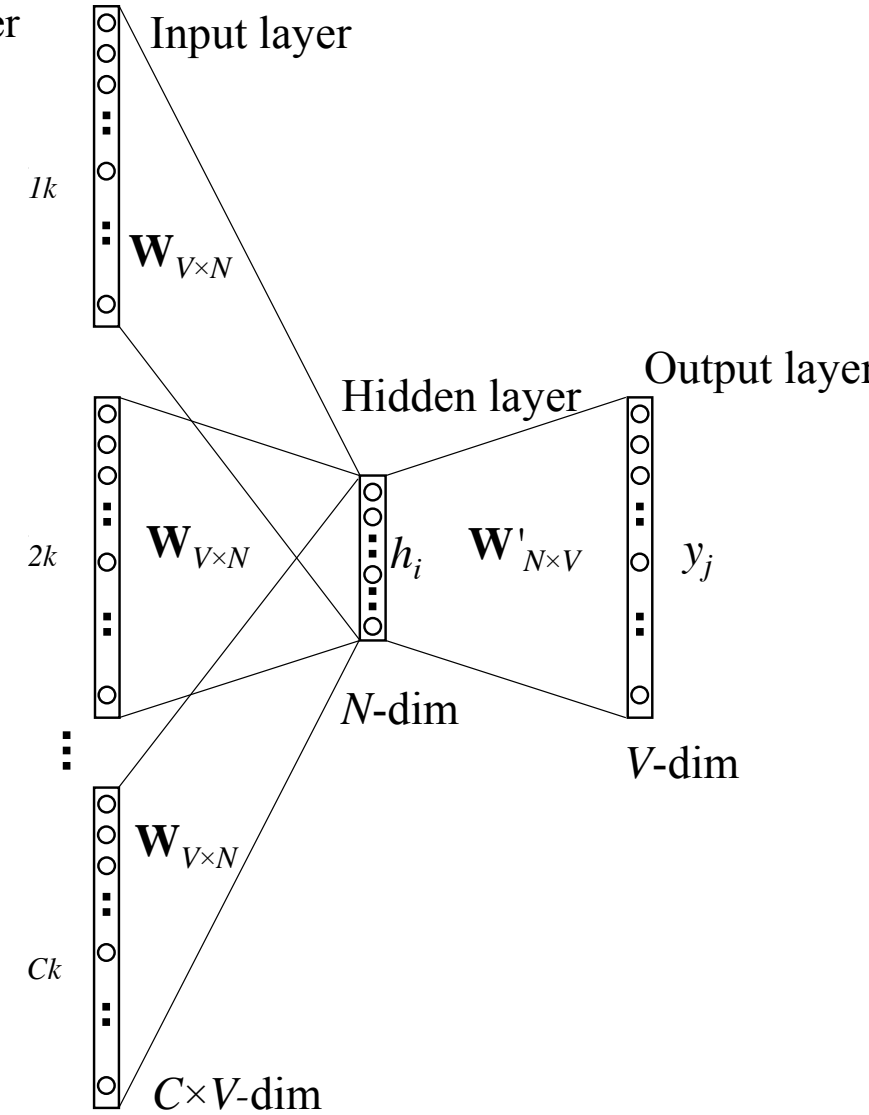
The products between the vectors can be calculated using a multi-layer neural network

In word2vec, the weights are trained using standard backpropagation

CBOW is significantly faster (because most of the computation is required in the second layer), but Skip-Gram is more accurate (CBOW embeddings are averaged)



Skip-Gram



CBOW

VU VRIJE UNIVERSITEIT AMSTERDAM

BERT (Bidirectional Encoder Representations from Transformers)

BERT is a more recent language model designed by Google in 2018 to perform many NLP tasks [1]

- Significantly more complex than word2vec
- Designed for **pre-training;** some extra work is needed before it can be used for a specific task
- Changed the field: “in a little over a year, BERT has become a ubiquitous baseline in NLP experiments", counting over 150 research publications analyzing and improving the model” [2]

VU VRIJE UNIVERSITEIT AMSTERDAM

[1] Devlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". arXiv:1810.04805v2
[2] Rogers, Anna; Kovaleva, Olga; Rumshisky, Anna (2020). "A Primer in BERTology: What We Know About How BERT Works"

BERT

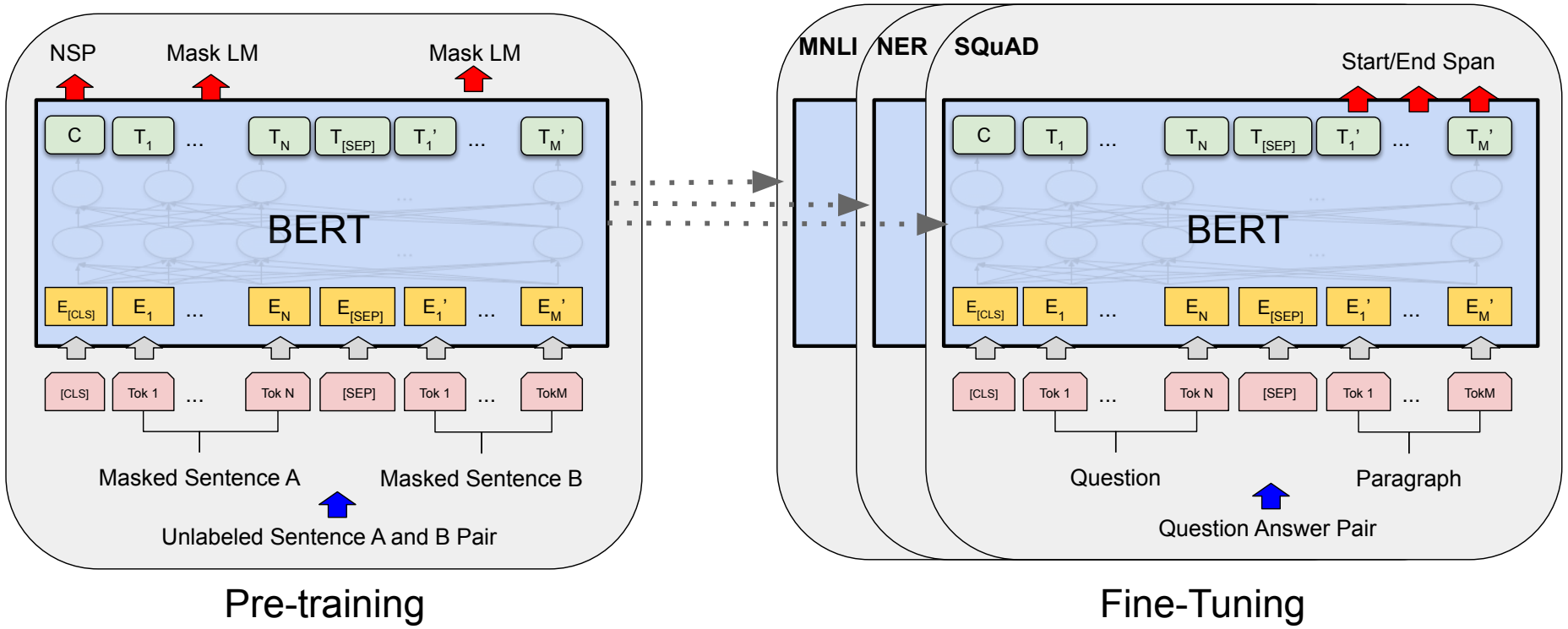
- BERT consists only of one encoder implemented with a transformer
- It has been pre-trained by Google consider all the content of Wikipedia and BooksCorpus
  - Task 1: Language Modeling
  - Task 2: Next Sentence Prediction

- BERT can be **fine-tuned** for some specific tasks
  - Single sentence classification
  - Question answering
  - Sentence tagging
  - Sentiment Analysis

BERT

Expensive, done by Google

Cheap, can be done by any of us



- Two variants:
  - BERT base – 12 layers, 12 attention heads and **110 million parameters**
  - BERT Large – 24 layers, 16 attention heads and **340 million parameters**

VU VRIJE UNIVERSITEIT AMSTERDAM

BERT

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

BERT is a language model that is general enough to perform well on many different tasks. What kind of knowledge does it contain?

It is not entirely clear why BERT works so well

VU VRIJE UNIVERSITEIT AMSTERDAM

https://github.com/google-research/bert

GPT

- Language model proposed by OpenAI in 2018
- Conceptually very similar to BERT (both use Transformers, but BERT uses an encoder while GPT uses a decoder)
- Like BERT, pre-training learns to predict the next word in a sequence
- Fine-tuning adapts the model for different tasks

Radford, Alec and Karthik Narasimhan. “Improving Language Understanding by Generative Pre-Training.” (2018).

GPT-2

- In 2019, OpenAI released GPT-2
- GPT-2 is significantly larger than BERT (1.5B parameters vs 340M parameters of BERT)
- Minimal (or no) fine-tuning
- **Translation:** Gives an "<*english*> = <*french*>" sentence as . Then provides "<english> = " and selects the most probable sequence
- **Summarization:** Provide "<article> TL;DR: <summary>". Then provides "<article>TL;DR: " and select the top sequence of 100 words
- Model outperforms BERT in many tasks

Radford, Alec et al. "Language Models are Unsupervised Multitask Learners." (2019).

GPT-3

- In 2020, OpenAI released the latest version of GPT
- Model contains **175 billion parameters**: (GPT-2 1.5 billions, BERT 340M)
- Source code, models are not available; Microsoft has exclusive rights to use the model
- Performance is truly impressive, GPT-3 can produce text that is indistinguisable from the one written by a human. For examples, see original paper (LINK)

Brown, Tom, et al. "Language models are few-shot learners." *Advances in neural information processing systems* 33 (2020): 1877-1901.

DALL-E and stable diffusion

- DALL-E 2 is a text-to-image model developed by OpenAI. Uses GPT-3 to interpret the input and generates the image. Open since September 2022 (https://openai.com/dall-e-2/)
- Stable Diffusion is another text-to-image model released in 2022 (https://stability.ai/blog/stable-diffusion-announcement). Source code is publicly available (demo)

Ramesh, Aditya, et al. "Hierarchical text-conditional image generation with clip latents." *arXiv preprint arXiv:2204.06125* (2022).

DALL-E 2 and stable diffusion

- “Portrait of a girl in a coffeeshop, reading a book, dramatic lighting”

VU VRIJE UNIVERSITEIT AMSTERDAM

Example taken from https://medium.com/mlearning-ai/dall-e2-vs-stable-diffusion-same-prompt-different-results-e795c84adc56, photos by https://jimclydemonge.medium.com/