CE6146
Introduction to Deep Learning
Transformers and Attention Mechanisms

Chia-Ru Chung

Department of Computer Science and Information Engineering

National Central University

2023/11/30

Outline

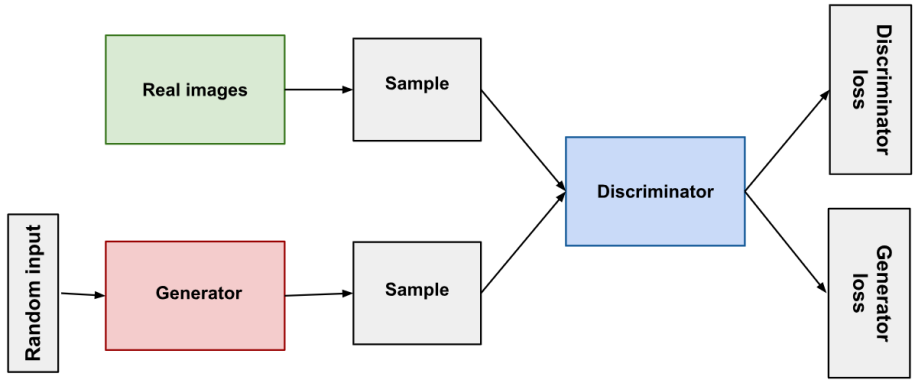
- Generative Adversarial Networks
- Introduction to AWS for Deep Learning
- Natural Language Processing
- Transformers and Attention Mechanisms

Generative Adversarial Networks

- **Brief Review**

What is a GAN

- A Generative Adversarial Network (GAN) is a class of machine learning systems where two neural networks contest with each other in a game.
- Conceived as a generative model, it's an approach to unsupervised learning.
- A GAN consists of two parts:
  - Generator: This network generates new data instances.
  - Discriminator: This network evaluates them.



Source: https://developers.google.com/machine-learning/gan/gan_structure?hl=zh-tw

Data Flow in GAN

1. Starting Point

   The Generator begins with a set of random noise (usually a normal distribution).

2. Data Generation

   This noise is input to the Generator, which transforms it into a data instance.

3. Data Evaluation

   The Discriminator takes in both real data (from the actual dataset) and fake data (from the Generator) and tries to distinguish between the two.

How to Train a GAN

1. Train the Discriminator
   - Provide it with real data and fake data from the Generator.
   - Update the Discriminator's weights based on its ability to correctly classify the real and fake data.
2. Train the Generator
   - Generate new data.
   - Pass it to the Discriminator. If the Discriminator classifies the output as real, then the Generator is doing well.
   - Update the Generator's weights based on the Discriminator’s response.
3. Iterate
   - Repeat this process until the Generator produces realistic data. The ideal endpoint is when the Discriminator can no longer reliably distinguish fake data from real data.

GAN Training

- Because a GAN contains two separately trained networks, its training algorithm must address two complications:
  - GANs must juggle two different kinds of training (generator and discriminator).
  - GAN convergence is hard to identify.

Source: https://developers.google.com/machine-learning/gan/training

Alternating Training (1/2)

- GAN training proceeds in alternating periods:
  - The discriminator trains for one or more epochs.
  - The generator trains for one or more epochs.
  - Repeat steps 1 and 2 to continue to train the generator and discriminator networks.
- As the generator improves with training, the discriminator performance gets worse because the discriminator can't easily tell the difference between real and fake.

Source: https://developers.google.com/machine-learning/gan/training

Alternating Training (2/2)

- This progression poses a problem for convergence of the GAN as a whole: the discriminator feedback gets less meaningful over time.
- If the GAN continues training past the point when the discriminator is giving completely random feedback, then the generator starts to train on junk feedback, and its own quality may collapse.
- For a GAN, convergence is often a fleeting, rather than stable, state.

Source: https://developers.google.com/machine-learning/gan/training

Loss Functions

- GANs try to replicate a probability distribution.
- They should therefore use loss functions that reflect the distance between the distribution of the data generated by the GAN and the distribution of the real data.
- Two common GAN loss functions:
  - minimax loss
  - Wasserstein loss

Source: https://developers.google.com/machine-learning/gan/loss

Minimax Loss (1/2)

- The generator tries to minimize the following function while the discriminator tries to maximize it: $\mathbb{E}_x[\log D(x)] + \mathbb{E}_z[\log(1 - D(G(z))]$
  - D(x) is the discriminator's estimate of the probability that real data instance x is real.
  - E_x is the expected value over all real data instances.
  - G(z) is the generator's output when given noise z.
  - D(G(z)) is the discriminator's estimate of the probability that a fake instance is real.
  - E_z is the expected value over all random inputs to the generator (in effect, the expected value over all generated fake instances G(z)).
  - The formula derives from the cross-entropy between the real and generated distributions.

Source: https://developers.google.com/machine-learning/gan/loss

Minimax Loss (2/2)

- The generator can't directly affect the log(D(x)) term in the function, so, for the generator, minimizing the loss is equivalent to minimizing log(1 - D(G(z))).

Source: https://developers.google.com/machine-learning/gan/loss

Wasserstein Loss

- This loss function depends on a modification of the GAN scheme (called "Wasserstein GAN" or "WGAN") in which the discriminator does not actually classify instances.
- Because WGAN can't really discriminate between real and fake, the WGAN discriminator is actually called a "critic" instead of a "discriminator".
- Critic Loss: D(x) - D(G(z))

  The discriminator tries to maximize this function. In other words, it tries to maximize the difference between its output on real instances and its output on fake instances.

- Generator Loss: D(G(z))

  The generator tries to maximize this function. In other words, It tries to maximize the discriminator's output for its fake instances.

Source: https://developers.google.com/machine-learning/gan/loss

Introduction to AWS for Deep Learning

- Introduction to Cloud Computing and AWS
- AWS Core Services for Deep Learning
- Deep Learning Frameworks on AWS

Cloud Computing

- Cloud computing is the on-demand delivery of compute power, database storage, applications, and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing.

Why Cloud Computing

- In the context of deep learning and research, cloud computing provides the necessary computational power and storage required to process and analyze large datasets, train complex models, and perform extensive simulations.
- It enables researchers and practitioners to access state-of-the-art GPUs and other computing resources that might be too costly or impractical to maintain on-premises.
- This democratizes access to advanced computational capabilities, allowing for more innovation and exploration in various fields.

How Does Cloud Computing Work (1/2)

- Imagine you need a book to read. You have two options: buy the book and bring it home (which is like owning a computer with software installed) or go to a public library to read it (which is like using cloud computing).
  - Buying a Book: It's yours, but it costs more, and you need space at home. If you want ten books, you need to buy all ten.
  - Using the Library: You can read any book you want, you don't pay for each book, and you don't need space at home for a big bookshelf. You just need a library card, which is like your cloud account.

How Does Cloud Computing Work (2/2)

- User Request: You (the user) send a request to access computing services. This is like asking the librarian for a book.
- Data Center Response: The cloud provider's data center (a massive collection of servers and storage) receives your request and processes it. This is like the librarian checking the book out for you.
- Resource Allocation: The cloud provider allocates resources to meet your request. This can include storage space, CPU, memory, etc., just like the librarian finds you a place to sit and read.
- Service Delivery: The requested service (like a software application or data storage) is delivered to your device over the internet, ready for use. This is like reading the book in the library.
- Pay for Use: You only pay for the cloud services you use, for the time you use them, just as you might pay a fee to the library if you use a special service or rent a book.
- Release Resources: When you're done, the resources you used are freed up for someone else to use. This is like returning the book so the next person can read it.

Types of Cloud Services

- Infrastructure as a Service (IaaS):

  Provides basic infrastructure, compute, networking, and storage resources on-demand.

- Platform as a Service (PaaS):

  Offers a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure.

- Software as a Service (SaaS):

  Delivers software applications over the internet, on-demand, and typically on a subscription basis.

Infrastructure as a Service (IaaS)

- Components of IaaS: Virtual machines, servers, storage, networks, and operating systems.
- Provider Responsibility: The cloud provider manages the infrastructure, while users manage the applications, data, runtime, and middleware.
- User Control: Users have control over the infrastructure without physically managing it.
- Use Cases: Test and development, website hosting, storage and backup, web apps, high-performance computing.
- Examples: AWS EC2, Google Compute Engine, Microsoft Azure.

Platform as a Service (PaaS)

- Components of PaaS: Hosted development kits, database management systems, business analytics.
- Provider Responsibility: Cloud providers manage the infrastructure and platforms that run the applications.
- User Control: Users manage the applications and services they develop, and the cloud provider manages everything else.
- Use Cases: Development framework, analytics or business intelligence, additional services like workflow, directory, security.
- Examples: AWS Elastic Beanstalk, Google App Engine, Microsoft Azure App Services.

Platform as a Service (PaaS)

- Components of SaaS: Software applications accessible from various devices over the internet or a thin client.
- Provider Responsibility: SaaS providers manage the infrastructure, middleware, app software, and app data.
- User Control: Users get to use the software without worrying about installation, maintenance, or coding.
- Use Cases: Email, calendaring, office tools, customer relationship management (CRM).
- Examples: Google Workspace, Salesforce, Microsoft Office 365.

Benefits for Deep Learning

- IaaS Benefits:

  Flexibility to choose the computing resources as per the need of the deep learning models, which can be scaled up as the complexity of the model increases.

- PaaS Benefits:

  Provides a platform with built-in software components and tools specifically designed for application development, which can be beneficial for developing custom deep learning models and managing their lifecycle.

- SaaS Benefits:

  Access to sophisticated applications with AI and machine learning capabilities without the need for a deep understanding of the underlying algorithms or infrastructure.

Cloud Deployment Models

- Public Cloud:

  Services are delivered over the public internet and shared across organizations.

- Private Cloud:

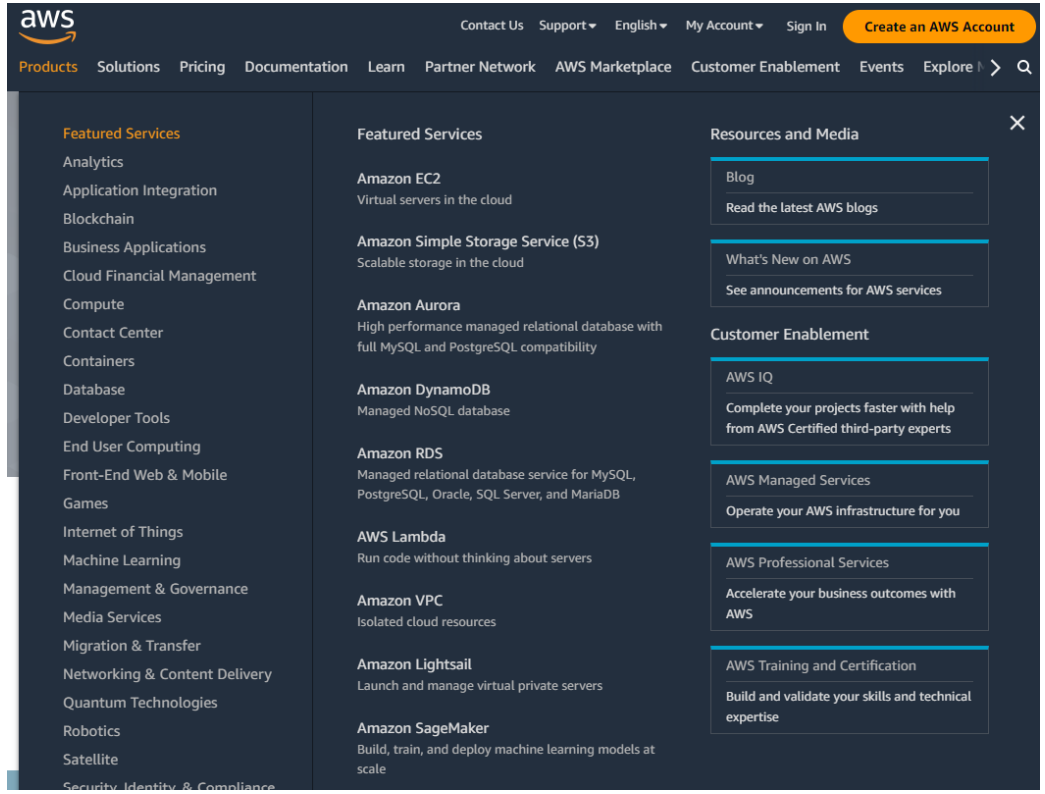
  Services are maintained on a private network for a single organization.

- Hybrid Cloud:

  Combines public and private clouds, bound together by technology that allows data and applications to be shared between them.

Overview of Amazon Web Services

- Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally.



Source: https://aws.amazon.com/tw/

Core AWS Services for Deep Learning (1/2)

- Amazon EC2: Provides scalable computing capacity in the AWS cloud. It reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. (https://aws.amazon.com/tw/ec2/)
- Amazon S3: Offers secure, durable, and highly-scalable object storage. It's a simple storage service that offers an extremely durable, highly available, and infinitely scalable data storage infrastructure at very low costs. (https://aws.amazon.com/tw/s3/)

Core AWS Services for Deep Learning (2/2)

- AWS Lambda: A serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you. (https://aws.amazon.com/tw/lambda/)
- Amazon SageMaker: A fully managed service that provides every developer and data scientist with the ability to build, train, and deploy machine learning models quickly. (https://aws.amazon.com/tw/sagemaker/)

AWS for Machine Learning and Deep Learning

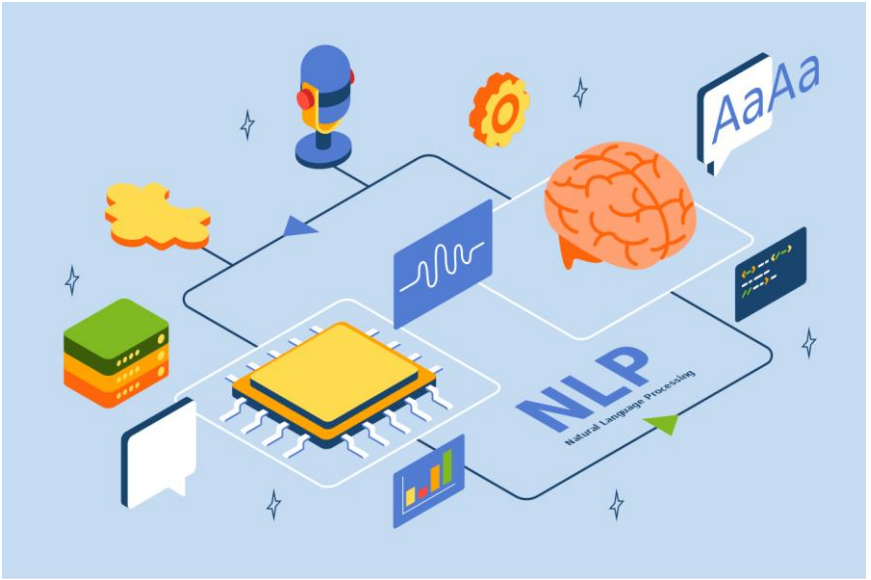
- Deep Learning AMIs: Amazon Machine Images (AMIs) designed for deep learning. Pre-installed with popular frameworks like TensorFlow, PyTorch, Apache MXNet, and others.
- Deep Learning Containers: AWS offers Docker images pre-installed with deep learning frameworks for use on AWS services like Amazon ECS and AWS Fargate.
- Elastic Inference: Allows you to attach low-cost inference acceleration to Amazon EC2 and SageMaker instances or ECS tasks, reducing the cost of inference without compromising performance.

Natural Language Processing

- Introduction
- Basics of NLP
- Deep Learning in NLP

Importance of NLP in the Context of AI

- Natural Language Processing (NLP) is a field at the intersection of computer science, artificial intelligence (AI), and linguistics. It focuses on enabling computers to understand, interpret, and respond to human language in a meaningful way.
- NLP involves the application of algorithms and models to process, analyze, and generate human language. This encompasses a range of tasks such as speech recognition, natural language understanding, natural language generation, and machine translation.



Source: https://en.wikipedia.org/wiki/Natural_language_processing

Why is NLP Important in AI

- Enhancing Human-Computer Interaction

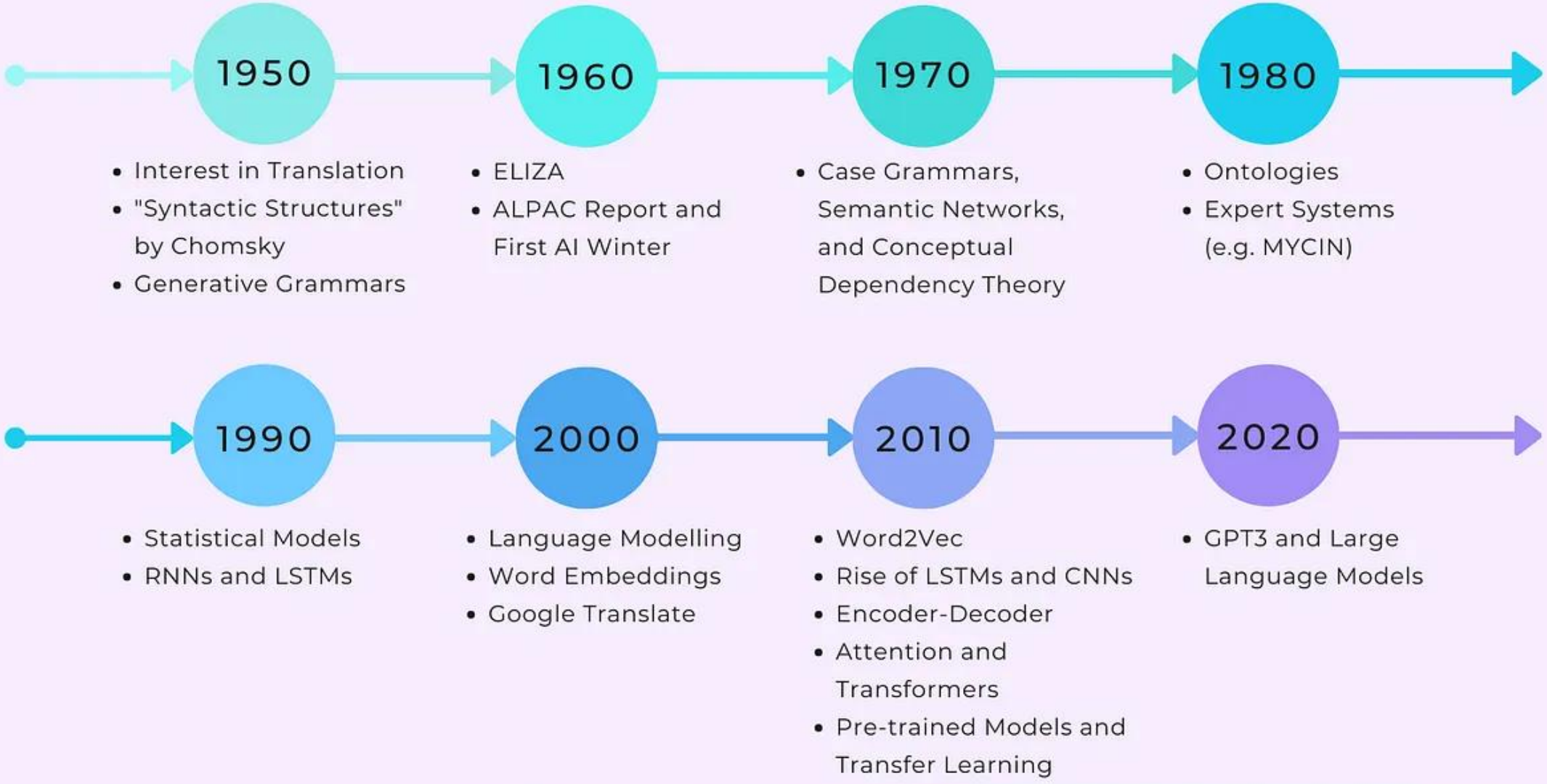
  NLP allows for more natural, conversational interfaces between humans and machines, enhancing user experience.
- Data Analysis and Insight Generation

  With the explosion of data in the form of text, NLP helps in extracting insights, summarizing content, and automating data-driven decision-making processes.
- Language Accessibility and Inclusion

  NLP technologies like translation services and voice recognition software make information accessible across language barriers, aiding in global communication and inclusion.
- Cross-Disciplinary Applications

  NLP finds its use in various fields such as healthcare for patient data analysis, in finance for sentiment analysis of markets, in law for document review, and in customer service for chatbots.

A Brief Timeline of NLP



Source: https://medium.com/nlplanet/a-brief-timeline-of-nlp-bc45b640f07d

Key Concepts in NLP (1/5)

- Tokenization
  - Breaking down text into smaller units (tokens), which can be words, phrases, or symbols.
  - It is the first step in preprocessing and is crucial for understanding the structure of the text.

Source: https://medium.com/@utkarsh.kant/tokenization-a-complete-guide-3f2dd56c0682

Key Concepts in NLP (2/5)

- Stemming and Lemmatization
  - Both techniques aim to reduce words to their base or root form.
  - Stemming is a heuristic process that chops off the ends of words, often leading to incorrect meanings and spelling.

    e.g., "fishing", "fished", "fisher" all reduce to "fish"
  - Lemmatization involves using vocabulary and morphological analysis, leading to more accurate results .

    e.g., "better" becomes "good"

Key Concepts in NLP (3/5)

- Part-of-Speech (POS) Tagging
  - This involves identifying parts of speech (verbs, nouns, adjectives, etc.) in a given sentence. It's crucial for understanding the grammatical structure and meaning of sentences.
  - Example: In "The quick brown fox jumps", "quick" and "brown" are tagged as adjectives, while "jumps" is a verb.

Source: https://medium.com/programming-with-data/9-%E8%A9%9E%E6%80%A7%E6%A8%99%E8%A8%BB-part-of-speech-tagging-dab799fdf75d

Key Concepts in NLP (4/5)

- Named Entity Recognition (NER)
  - NER identifies and classifies named entities in text into predefined categories like names of persons, organizations, locations, expressions of times, quantities, monetary values, etc.
  - Example: In "Apple Inc. was founded by Steve Jobs", "Apple Inc." is recognized as an organization, and "Steve Jobs" as a person.

Key Concepts in NLP (5/5)

- Bag of Words (BoW) and TF-IDF
  - BoW is a simple feature extraction technique used in NLP and information retrieval. In this model, text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.
  - TF-IDF (Term Frequency-Inverse Document Frequency): This statistical measure evaluates how relevant a word is to a document in a collection of documents. It's used as a weighting factor in information retrieval and text mining.

What Are Word Embeddings

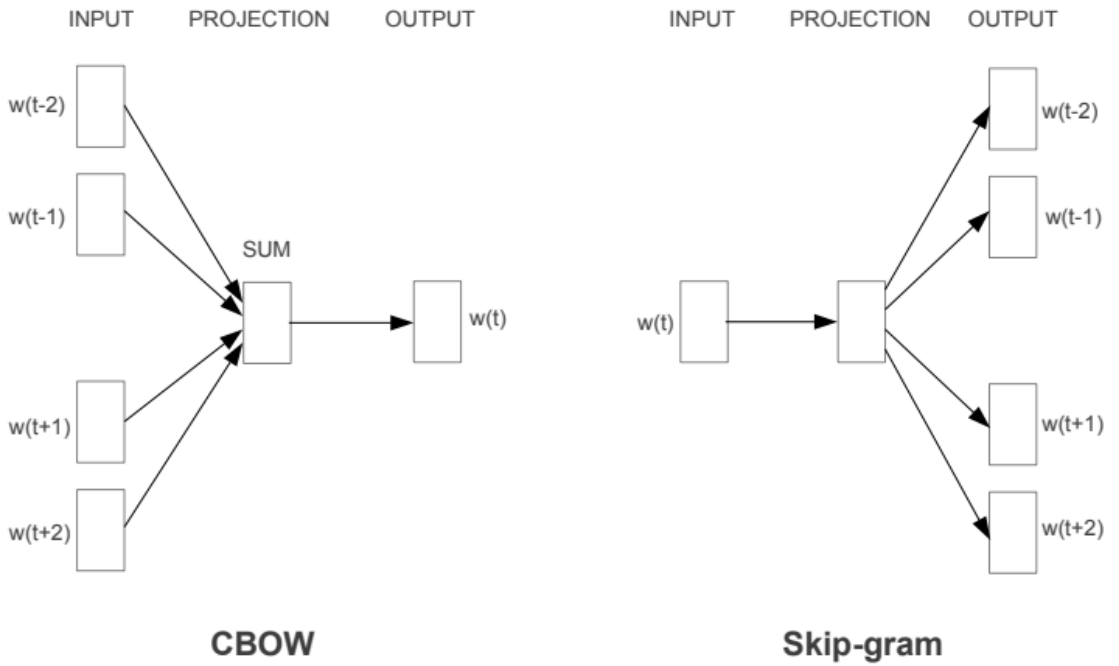
- Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. They are a distributed representation for text that is perhaps one of the key breakthroughs for the impressive performance of deep learning methods on challenging NLP problems.
- They are a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector, and the vector values are learned in a way that resembles a neural network.

Importance of Word Embeddings

- They capture not just the semantic meaning of individual words, but also the different nuances and contexts in which they are used. This leads to a much more nuanced understanding of language by NLP models.
- They enable NLP models to understand synonyms and antonyms, interpret words with multiple meanings based on context, and even perform analogical reasoning (e.g., "king" is to "queen" as "man" is to "woman").

Key Types of Word Embeddings (1/2)

- Word2Vec
  - Developed by Google, Word2Vec uses Neural Networks to learn word associations from a large corpus of text.
  - Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence.
  - There are two architectures Word2Vec uses: Continuous Bag of Words (CBOW) and Skip-gram.



Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Key Types of Word Embeddings (2/2)

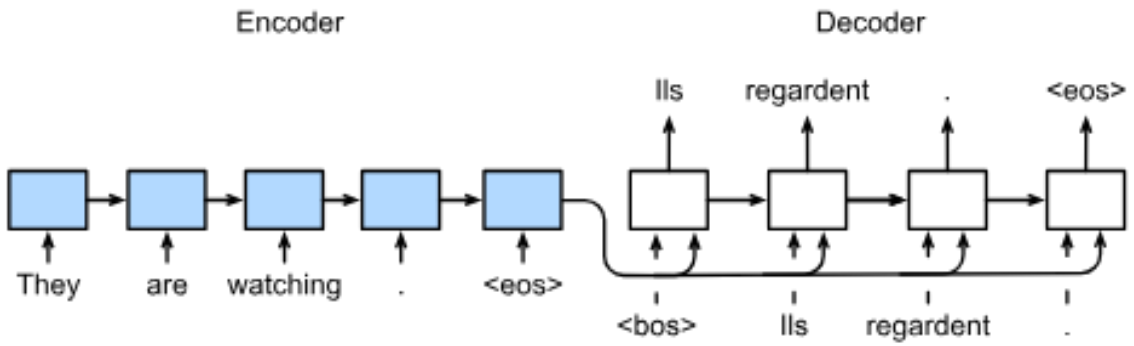
- GloVe (Global Vectors for Word Representation)
  - Developed by Stanford, GloVe is an unsupervised learning algorithm for obtaining vector representations for words.
  - This method is based on aggregating the global word-word co-occurrence matrix from a corpus, and then performing matrix factorization.

Word2Vec vs GloVe

- Word2Vec focuses on local context, learning from neighboring words, whereas GloVe integrates global word co-occurrence statistics, capturing broader contextual information.
- Both capture semantic relationships, but GloVe's use of global statistics often allows it to capture more subtle meanings and relationships.
- Word2Vec is generally more efficient with large datasets, but GloVe requires a significant amount of memory for its co-occurrence matrix.
- Both models face challenges with OOV words, as their embeddings are fixed post-training.
- While both are widely used in various NLP tasks, the choice between them may depend on the specific requirements of the task, such as the need for capturing local versus global context, or the availability of computational resources.

What are Seq2Seq Models

- Sequence-to-Sequence (Seq2Seq) models are a type of neural network architecture used to convert sequences from one domain (e.g., a sentence in English) to sequences in another domain (e.g., the same sentence translated into French).



Source: https://d2l.ai/chapter_recurrent-modern/seq2seq.html

Transformers and Attention Mechanisms

- Attention Mechanisms
- Transformers

What are Attention Mechanisms

- Attention mechanisms in deep learning are inspired by the human process of focusing on a particular part of the information while ignoring others.
- In the context of neural networks, they enable the model to dynamically focus on certain parts of the input data that are most relevant to the task at hand, rather than processing the entire input in a uniform manner.

Why are Attention Mechanisms Important

- Handling Long Sequences: They allow models to handle long input sequences effectively, overcoming the limitations of traditional sequence models like RNNs and LSTMs, which struggle with long-range dependencies.
- Improved Contextual Understanding: Attention mechanisms provide a way for models to capture contextual relationships in data, essential for tasks like language translation and text summarization.
- Increased Model Efficiency: By focusing on relevant parts of the input, attention mechanisms can reduce computational load and improve the efficiency of the model.

How do Attention Mechanisms Work

- Components: The key components of an attention mechanism are Queries (Q), Keys (K), and Values (V).
- Process:
  1) The model generates a Query for each output element.
  2) Each Query is compared against all Keys (usually using dot product) to produce attention scores.
  3) These scores are normalized (often using softmax) to form a probability distribution, known as attention weights.
  4) The model computes a weighted sum of the Values using these attention weights, producing the output that is contextually focused on relevant parts of the input.

What are Transformers

- The Transformer is a type of deep learning model introduced in the paper "Attention Is All You Need" by Vaswani et al.
- The model avoids recurrence, commonly found in RNNs, and instead relies solely on attention mechanisms to establish global dependencies between input and output.
  - Earlier models like RNNs and LSTMs process sequential data (such as text or time-series data) using recurrent layers. These layers work through the sequence one element at a time, maintaining a hidden state that theoretically captures information from all previous elements in the sequence.
  - This sequential processing limits the ability to parallelize operations, which can make training slower, especially for long sequences. Additionally, recurrent layers often struggle with long-range dependencies due to issues like vanishing gradients.

Why are Transformers Important

- Parallel Processing: Unlike RNNs and LSTMs, Transformers process all elements of the input data in parallel, significantly reducing training time.
- Handling Long-Range Dependencies: Transformers, with their self-attention mechanism, are adept at handling long-range dependencies in the data, making them highly effective for complex NLP tasks.
- Scalability and Effectiveness: Transformers have been shown to scale well with data and model size, leading to state-of-the-art performance in various tasks like language translation, text generation, and more.

How do Transformers Work (1/2)

- Architecture: The Transformer model has two main parts: an encoder and a decoder. Each consists of multiple identical layers containing self-attention and fully connected feed-forward networks.
- Self-Attention: In the self-attention mechanism, all three components of attention (Queries, Keys, and Values) are derived from the same input. The model computes attention for every element in the input sequence with respect to every other element, enabling it to capture the context of the entire sequence.

How do Transformers Work (2/2)

- Positional Encoding: Since Transformers do not inherently process data sequentially, they use positional encodings to maintain the order of the sequence.
- The Decoder: The decoder also uses attention, looking at the encoder's output and its own output from previous time steps, to generate the final output sequence.

Thank you