Human and Machine Language

TIN0255/09P9M80/D82 - Aprendizagem Profunda

2026.2 - Prof. Pedro Moura

Human and Machine Language

- We saw an introduction to the high-level theory of deep learning via analogy to the biological visual system.
- One of the technique's core strengths lies in its ability to **learn features automatically from data**.
- Now we are going to examine how deep learning is incorporated into human language applications.
  - Particular emphasis on how it can automatically learn features that represent the **meaning** of words.

Human and Machine Language

- According to the austro-british philosopher Ludwig Wittgenstein, in his seminal work *Philosophical Investigations*:
- **“The meaning of a word is its use in the language.”**
- He further wrote: “One cannot guess how a word functions. One has to look at its use, and learn from that.”
- Words on their own have no real meaning → it is by their use within the **larger context of that language** that we’re able to ascertain their meaning.

Human and Machine Language

- As we will see in this class, Natural Language Processing (NLP) with deep learning relies heavily on this premise.
- The **word2vec** technique → converts words into numeric model inputs.
  - Explicitly derives its semantic representation of a word by analyzing it within its **contexts** across a large **body of language**.
- Let's begin by breaking down deep learning for NLP as a discipline.
- Then we go on to discuss modern deep learning techniques for representing words and language.

Deep Learning for Natural Language Processing

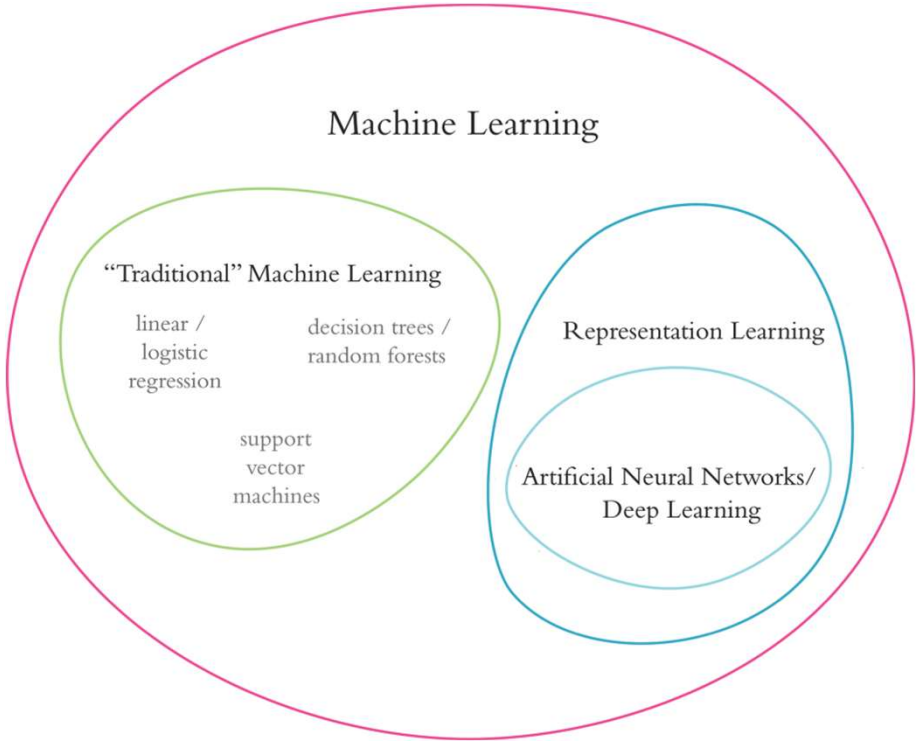
- Two core concepts for this class: **Deep Learning** and **Natural Language Processing**.
- Initially, we will cover these concepts separately, and then we weave them together.
- Deep Learning can be defined as the layering of simple algorithms called **artificial neurons** into several layers deep.
- Let's see how deep learning resides within the machine learning family of **Representation Learning** approaches.

Deep Learning for Natural Language Processing

- The representation learning family includes any techniques that **learn features from data automatically**.
  - Nowadays, deep learning dominates this family.

- The figure in next slide lays the foundation for understanding the advantage of representation learning relative to traditional machine learning approaches.

Deep Learning for Natural Language Processing



Venn diagram that distinguishes the traditional family from the representation learning family of machine learning techniques.

Deep Learning for Natural Language Processing

- Traditional ML works well because of clever, human-designed code that transforms raw data (images, audio of speech, or text from documents) into input features for machine learning algorithms.
  - Examples: regression, random forest, or support vector machines.
- The algorithms are adept at **weighting features** but not particularly good at learning features from raw data directly.
- This manual creation of features is often a **highly specialized task**.
  - For working with language data, for example, it might require graduate-level training in linguistics.

Deep Learning for Natural Language Processing

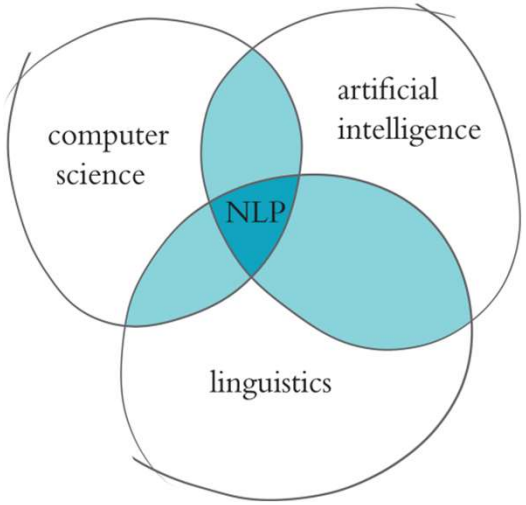
- Primary benefit of deep learning → it eases this requirement for subject-matter expertise.
- Instead of manually curating input features from raw data, one can **feed the data directly** into a deep learning model.
- Over the course of many examples → the artificial neurons of the first layer of the network learn how to **represent simple abstractions** of these data.
- Each successive layer learns to **represent increasingly complex nonlinear abstractions** on the layer that precedes it.
- **Convenience + additional advantages.**

Deep Learning for Natural Language Processing

- Features engineered by humans tend **not to be comprehensive and excessively specific**.
- It can also involve lengthy, ongoing loops of feature ideation, design, and validation that could **stretch for years**.
- **Representation learning models generate features quickly.**
  - Typically over hours or days of model training.
- **It adapts straightforwardly to changes in the data.**
  - Example: new words, meanings, or ways of using language.
- **It adapts automatically to shifts in the problem being solved.**

Natural Language Processing

- **Natural Language Processing** is a field of research that sits at the intersection of computer science, linguistics, and artificial intelligence.



NLP sits at the intersection of the fields of computer science, linguistics, and artificial intelligence.

Natural Language Processing

- NLP involves taking the **naturally spoken** or **naturally written language** of humans and **processing it with machines** to automatically complete some **task** or to make a **task easier** for a human to do.

- It does not correspond to natural language: code written in a software language or short strings of characters within a spreadsheet.

- Examples of NLP in industry:
  - **Classifying documents:** using the language within a document (e.g., a Tweet) to classify it into a particular category (e.g., positive sentiment).
  - **Machine translation:** assisting language-translation firms with machine-generated suggestions from a source language (e.g., English) to a target language (e.g., German); increasingly, fully automatic (not always perfect) between languages.
  - **Search engines:** autocompleting users' searches (besides **trie**) and predicting what information or website they're seeking.

Natural Language Processing

- **Speech recognition:** interpreting voice commands to provide information or take action, as with virtual assistants like Amazon's Alexa, Apple's Siri, or Microsoft's Cortana.
- **Chatbots:** carrying out a natural conversation for an extended period of time (though this is seldom done convincingly today, they are nevertheless helpful for relatively linear conversations on narrow topics such as customer-service phone calls).

- Some of the easiest NLP applications to build are spell checkers, synonym suggesters and keyword-search querying tools.
  - These tasks can be fairly straightforwardly solved with deterministic, rules-based code (e.g., reference dictionaries or thesauruses).

- Deep learning models are unnecessarily sophisticated for these applications.

Natural Language Processing

- Intermediate-complexity NLP tasks:
  - Assigning a school-grade reading level to a document;
  - Predicting the most likely next words while making a query in a search engine;
  - Classifying documents;
  - Extracting information like prices or named entities from documents or websites.

- **These intermediate NLP applications are well suited to solving with deep learning models.**

- Later in our classes, we'll leverage a variety of deep learning architectures to predict the sentiment of film reviews.

Natural Language Processing

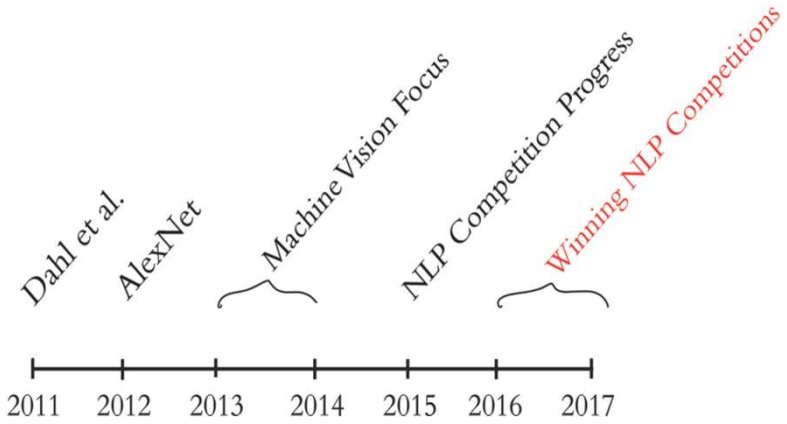
- The most sophisticated NLP implementations are required for machine translation, automated question-answering, and chatbots.

- These are tricky because:
  - They need to handle application-critical nuance;
  - A response to a question can depend on the intermediate responses to previous questions;
  - Meaning can be conveyed over the course of a lengthy passage of text consisting of many sentences.

- The content we cover in our classes will serve as a superb foundation for their development.

A Brief History of Deep Learning for NLP

- This timeline calls out recent milestones in the application of deep learning to NLP.



Milestones involving the application of deep learning to natural language processing.

A Brief History of Deep Learning for NLP

- It begins in 2011, when the University of Toronto computer scientist George Dahl and his colleagues at Microsoft Research revealed the first major breakthrough: **involving a deep learning algorithm applied to a large dataset.**

- This breakthrough happened to involve natural language data.

- Dahl and his team trained a deep neural network to recognize a substantial vocabulary of words from audio recordings of human speech.

- A year later, the next landmark also came out of Toronto: AlexNet blowing the traditional ImageNet competition.
  - This performance heralded a focus on applying deep learning to machine vision applications.

A Brief History of Deep Learning for NLP

- By 2015, the deep learning progress being made in machine vision began to spill over into NLP competitions.
  - Such as those that assess the accuracy of machine translations from one language into another.
- These deep learning models approached the precision of traditional machine learning approaches.
- **However, they required less research and development time while conveniently offering lower computational complexity.**
- Reduction in computational complexity → provided Microsoft the opportunity to squeeze real-time machine translation software onto mobile phone processors.

A Brief History of Deep Learning for NLP

- Remarkable progress for a task that previously had required an Internet connection and computationally expensive calculations on a remote server.

- In 2016 and 2017, deep learning models entered into NLP competitions not only were more efficient than traditional machine learning models, but **they also began outperforming them on accuracy.**

Computational Representations of Language

- In order for deep learning models to process language, we have to supply that language to the model in a way that it can digest.

- For all computer systems, this means a quantitative representation of language, such as a two-dimensional matrix of numerical values.

- Two popular methods for converting text into numbers are **one-hot encoding** and **word vectors**.
  - There also methods based on word frequency: TF-IDF (term frequency-inverse document frequency) and PMI (pointwise mutual information).

One-Hot Representations of Words

- The traditional approach to encoding natural language numerically for processing it with a machine is **one-hot encoding**.

The bat sat on the cat.

words						
the	1	0	0	0	1	0
bat	0	1	0	0	0	0
on	0	0	0	1	0	0
⋮						
n_{unique_words}						

One-hot encodings of words in the sentence.

One-Hot Representations of Words

- In this approach, the words of natural language in a sentence (e.g., “the,” “bat,” “sat,” “on,” “the,” and “cat”) are represented by **the columns of a matrix**.

- Each row in the matrix, meanwhile, represents a unique word.

- If there are 100 unique words across the *corpus* of documents you’re feeding, then your matrix of one-hot-encoded words will have 100 rows.
  - A *corpus* (from the Latin “body”) is the collection of all of the documents (the “body” of language) you use as your input data for a given natural language application.

One-Hot Representations of Words

- Cells within one-hot matrices consist of **binary values**, that is, they are a 0 or a 1.
- Each column contains at most a single 1, but is otherwise made up of 0s: **sparse matrices**.
- **Values of one indicate the presence of a particular word (row) at a particular position (column) within the corpus.**
- In the example, our entire corpus has only six words, five of which are unique.
- The first unique word - `the` - occurs in the first and fifth positions, as indicated by the cells containing 1s in the first row of the matrix.

One-Hot Representations of Words

- One-hot representation is **straightforward**, and they are an acceptable format for feeding into a deep learning model (or other machine learning model).

- Problem: **simplicity** and **sparsity**.
  - This is limiting when incorporated into a natural language application.

- **Consequence is a major problem: curse of dimensionality**.
  - Recurrent problem in Data Science/Artificial Intelligence.

- This describes a number of **unintuitive properties** that occur in high-dimensional Euclidean and non-Euclidean spaces.

One-Hot Representations of Words

- In high-dimensions: vectors are isolated points in a **vast and almost empty space**.

- Some properties that occur in these spaces:
  - Almost all pair of points are equally far away from one another (equidistants).

  - Almost any two vectors are almost orthogonal (inner product is equal to zero).

- **Clusterization and many other tasks are really hard in high dimensions!**
  - It is hard to find clusters among so many pairs that are all at approximately the same distance.

Word Vectors

- Vector representations of words are the information-dense alternative to one-hot encodings of words.
- One-hot representation → information about word location only.
- **Word vectors** (word embeddings or vector-space embeddings) → capture information about word meaning as well as location.
- **Key advantage:** word vectors enable deep learning NLP models to **automatically learn linguistic features**.

Word Vectors

- Using word vectors → we assign each word within a corpus to a meaningful location within a **multidimensional space** called the **vector space**.

- In contrast to one-hot encoding, word embeddings don't suffer from **curse of dimensionality**.

- The space is **much more restricted** and it is **easier** for a computer to **establish groupings** and to discover regularities there.

Word Vectors

- There are **two main approaches for learning word embeddings**, both relying on the contextual knowledge.

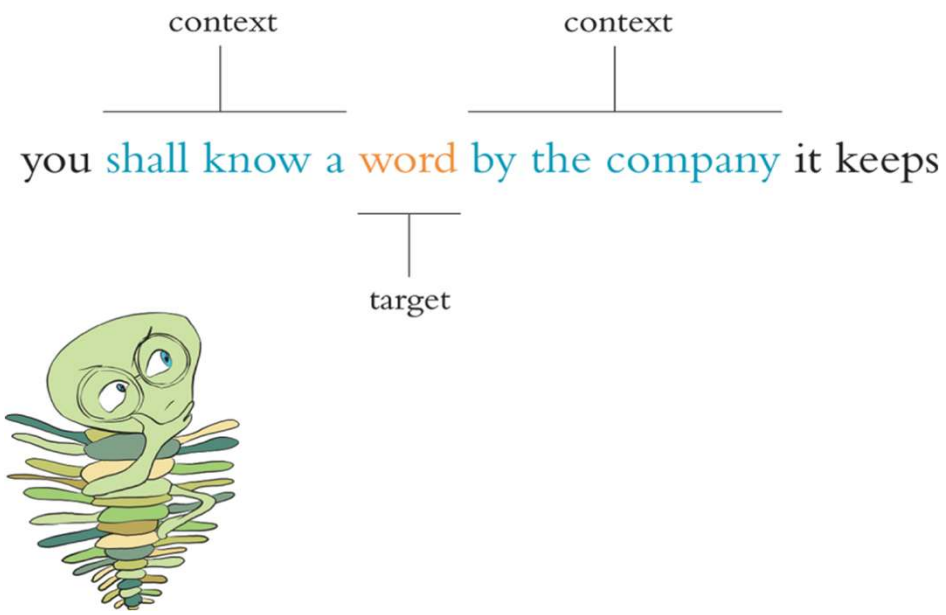
- **Count-based:** the first one is unsupervised, based on matrix factorization of a global word co-occurrence matrix.
  - Raw co-occurrence counts do not work well, so we want to do smart things on top.

- **Context-based:** the second approach is supervised. Given a local context, we want to design a model to predict the target words and in the meantime, this model learns the efficient word embedding representation.

Word Vectors

- Initially, each word is assigned to a random location within the vector space.
- By considering the words that tend to be **around a given word** → words within the vector space can be shifted into locations that represent the **meaning of the words**.
- This additional information renders word vectors favorable for a variety of reasons that we will see.

Word Vectors



A toy-sized example for demonstrating the high-level process behind techniques like word2vec and GloVe that convert natural language into word vectors.

Word Vectors

- Commencing at the first word, moving to the right one word at a time, we consider each word to be the **target word**.
  - In the example, the target word under consideration is `word`.
  - The next would be `by` and so on.
- For each target word, we consider it relative to the words around it: **context words**.
  - In our example, we use a context-word size window of three words.
- For the target word `word`, the three words to the left and to the right together constitue a **total of six context words**.
- When we move along to the subsequent target word (`by`), the windows of context words also shift one position to the right.
  - It then drops `shall` and `by` as context words while adding `word` and `it`.

Word Vectors

- Two of the most popular techniques for converting natural language into word vectors are **word2vec** and **GloVe**.

- After training over a large corpus → we gradually assign words that **tend to appear in similar contexts to similar locations** in vector space.

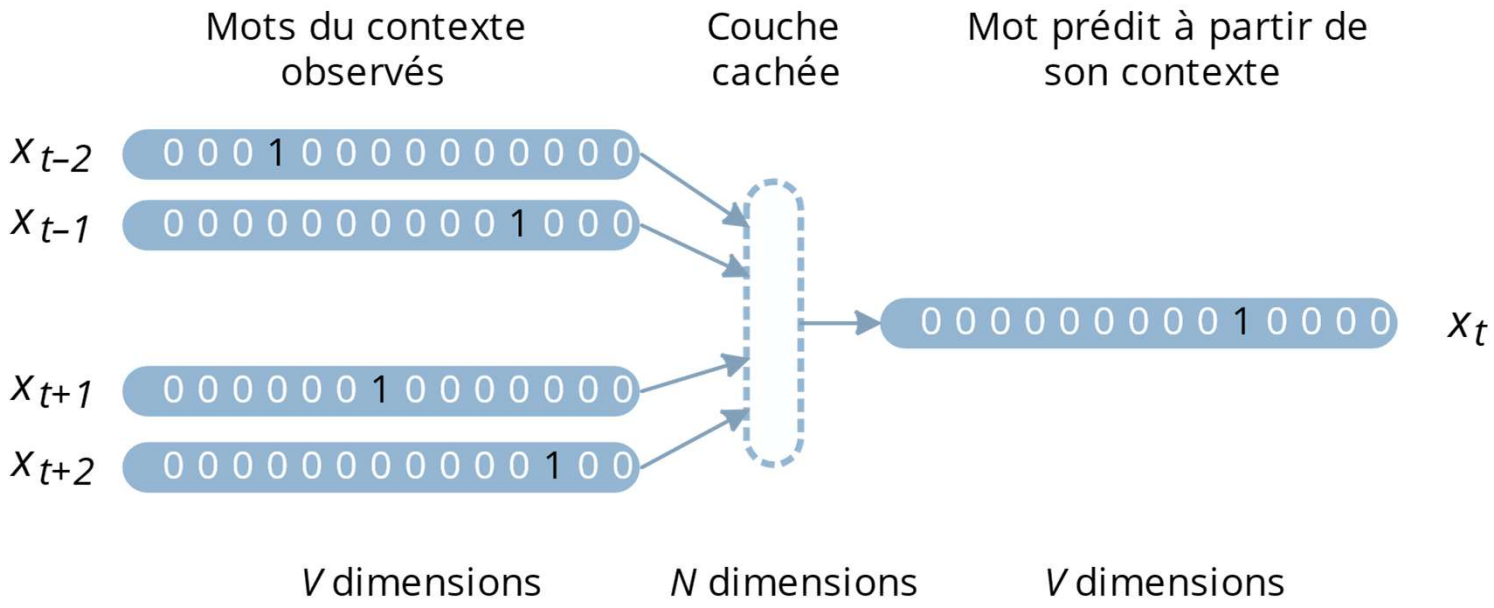
CBOW Model

- **Continuos Bag-of-Words (CBOW) Model**: objective is to **accurately predict the target word given its context words**.
- The model tries to **predict the *i*th word**, w_i, **given the context**

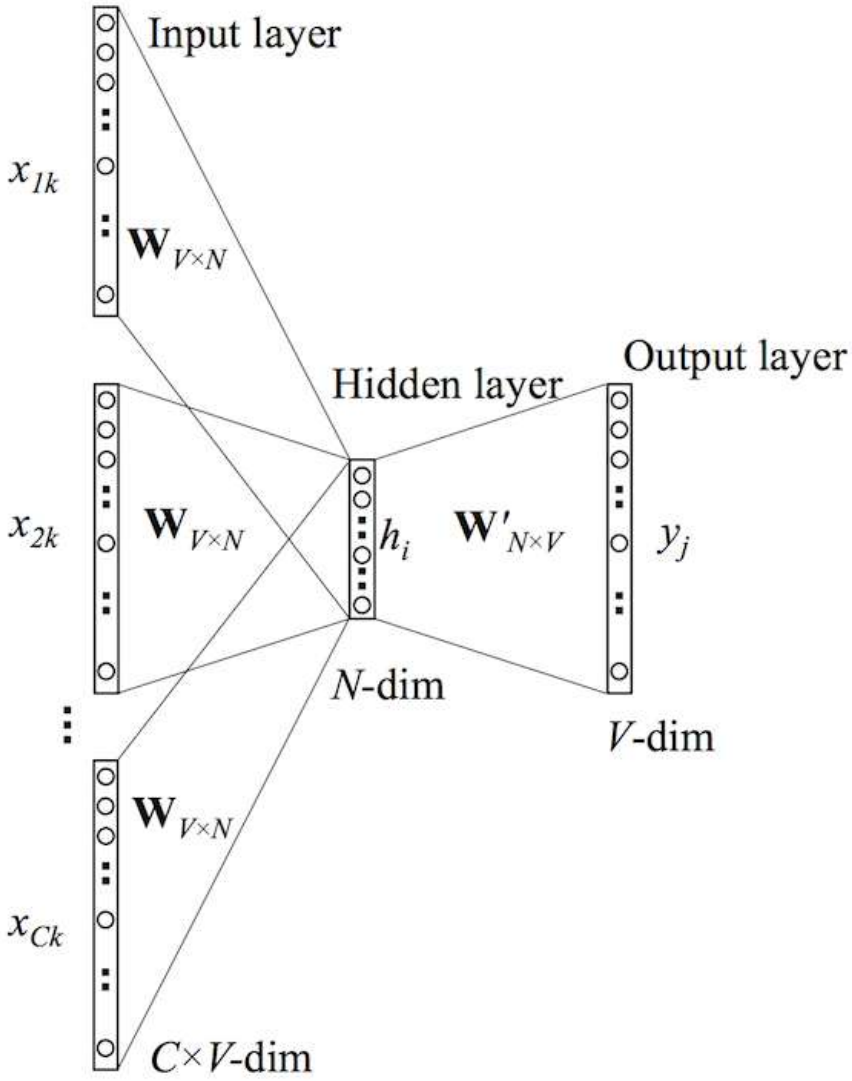
$$W_{i-t}W_{i-t+1} \ldots W_{i-1}W_{i+1} \ldots W_{i+t-1}W_{i+t}$$

- Total number of words in the context window is $m = 2t$.
- Input is the **one-hot encoded representation** of the context words.
  - Output is the *softmax* probability distribution.

CBOW Model


Mots du contexte observés
Couche cachée
Mot prédit à partir de son contexte
x_{t-2}
0 0 0 1 0 0 0 0 0 0 0 0 0 0
x_{t-1}
0 0 0 0 0 0 0 0 0 0 1 0 0 0
x_{t+1}
0 0 0 0 0 0 1 0 0 0 0 0 0 0
x_{t+2}
0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0
x_t
V dimensions
N dimensions
V dimensions

CBOW Model


Input layer
x_{1k}
$\mathbf{W}_{V\times N}$
Hidden layer
Output layer
x_{2k}
$\mathbf{W}_{V\times N}$
h_i
$\mathbf{W}'_{N\times V}$
y_j
N-dim
V-dim
$\mathbf{W}_{V\times N}$
x_{Ck}
$C\times V$-dim

Skip-gram Model

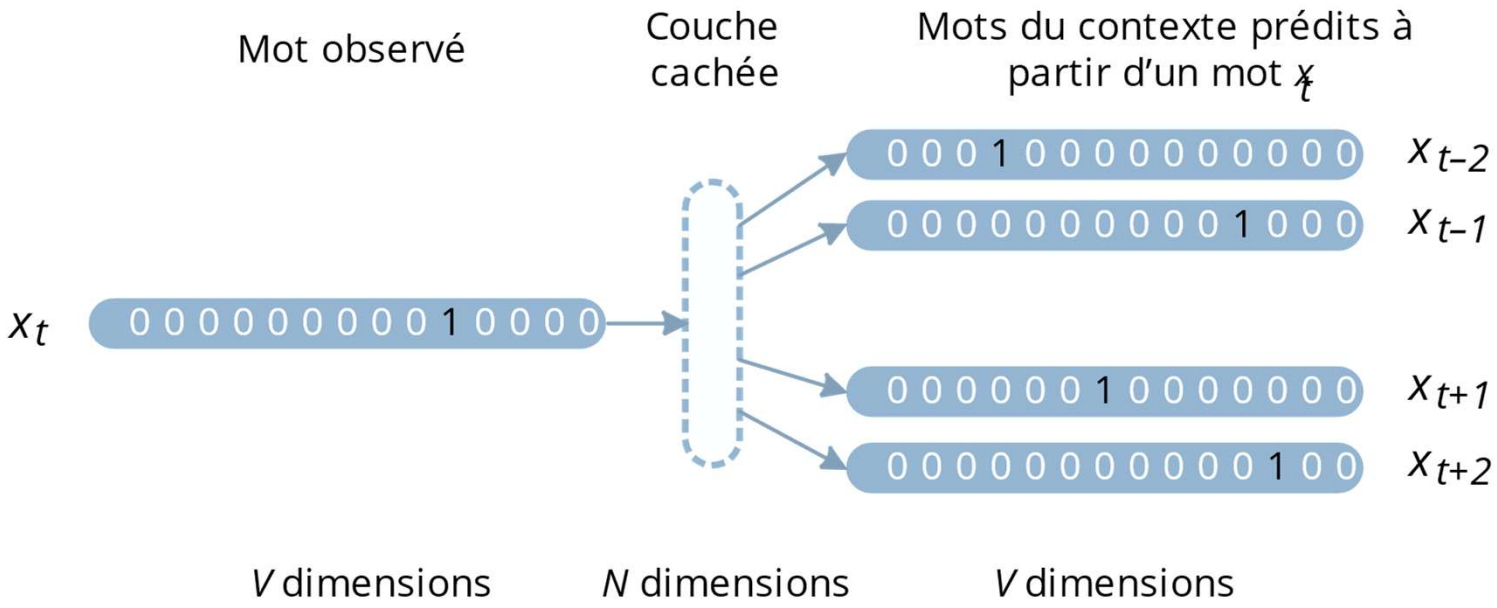
- **Alternatively, we could predict context words given a target word → Skip-Gram.**

- The model tries to **predict the context**

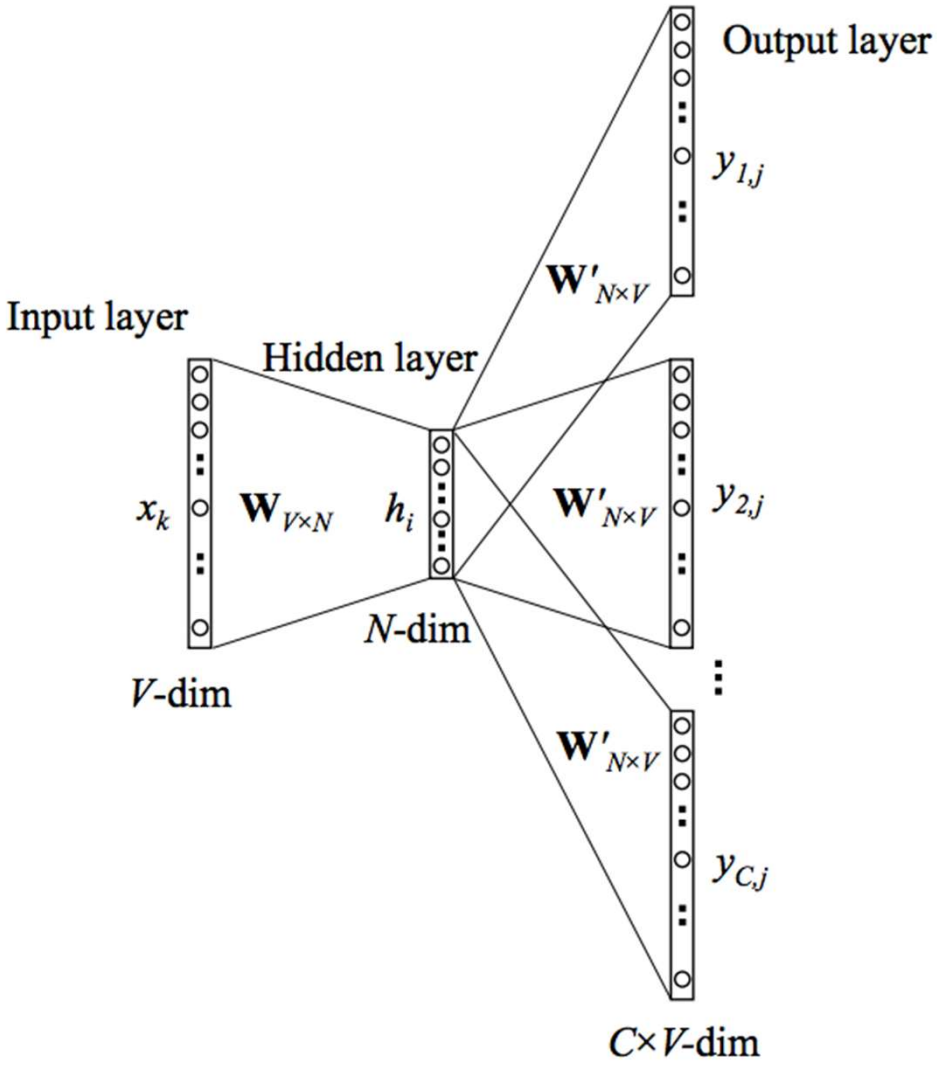
$$W_{i-t}W_{i-t+1} \ldots W_{i-1}W_{i+1} \ldots W_{i+t-1}W_{i+t}$$

given **the *i*th word**, W_i.

Skip-gram Model



Skip-gram Model



Word Vectors

- The space can have any number of dimensions: an ***n*-dimensional vector space**.

- In practice, depending on the richness of the corpus and the complexity of our NLP application → word-vector space could have dozens, hundreds, or thousands of dimensions.

- Any given word from our corpus (e.g. `king`) is assigned a location within the vector space.
  - In a 100-dimensional space → the word `king` is represented by a vector v_{king} of 100 components

Word Vectors

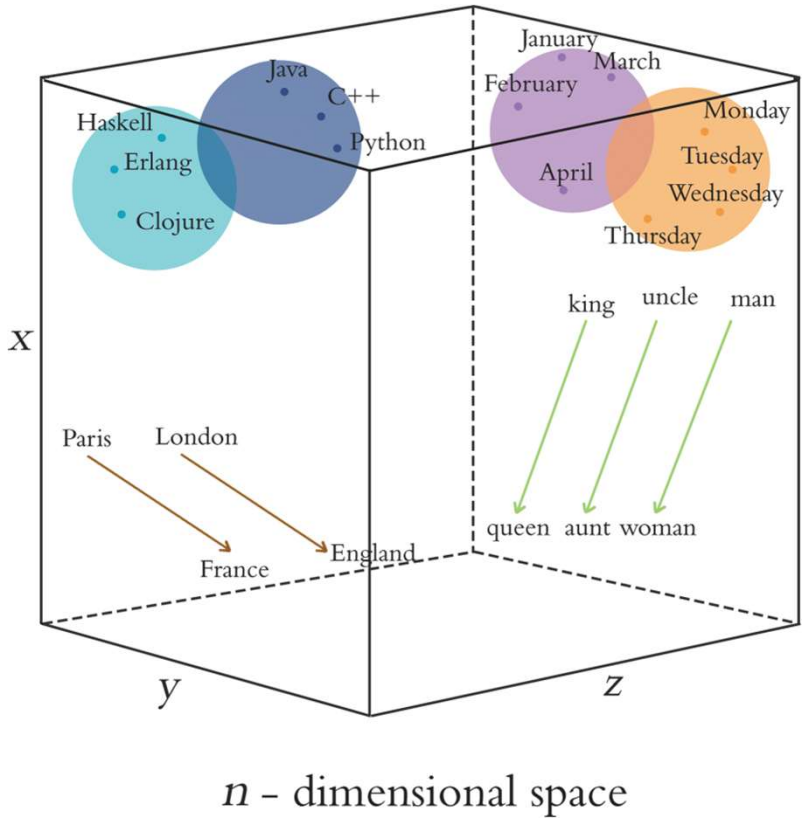


Diagram of word meaning as represented by a three-dimensional vector space.

Word Vectors

- In the last figure, any given word from our corpus needs three numeric coordinates to define its location within the vector space: *x*, *y* and *z*.
- The meaning of the word `king` is represented by a vector v_{king} that consists of three numbers.
- For example, if v_{king} is located at the coordinates $x = -0.9$, $y = 1.9$ and $z = 2.2$ in this vector space, we could use the annotation *[-0.9, 1.9, 2.2]* to describe it.
  - This succint annotation will come in handy shortly when we perform arithmetic operations.

Word Vectors

- The closer two words are within vector space, the closer their meaning, as determined by the similarity of the context words appearing near them.
  - Measured by **Euclidean distance**.
- Synonyms and common misspellings of a given word → nearly identical context words and nearly identical locations in vector space.
- Words that are used in similar contexts → tend to occur near each other in vector space.
  - Such as those that denote time.
- `Monday`, `Tuesday` and `Wednesday` are represented by orange-colored dots located within the orange days-of-the-week cluster.

Word Vectors

- **Months of the year might occur in their own purple cluster.**
  - **Adjacent to the cluster of days of the week.**
- They both relate to the date, but they are separate clusters within a broader *dates* region.
- Programming languages are clustered together in some location within the word-vector space.
  - Distant from time-denoting words.
- **`Java`, `C++` and `Python` form one sub-cluster → object-oriented programming languages.**
- **`Haskell`, `Clojure` and `Erlang` form a separate sub-cluster → functional programming languages.**

Word-Vector Arithmetic

- Particular movements across come to represent relative particular meanings between words.
- **This is a fantastic property!**
- In the image, brown arrows represent the relationship between countries and their capitals.
- Calculate the direction and distance between the coordinates for `Paris` and `France` → trace this from `London`, we should find the neighborhood of `England`.
- Calculate the direction and distance between the coordinates for `man` and `woman` → this movement trough vector space represents **gender**.

Word-Vector Arithmetic

- If we make this movement from any given male tem, we find our way to a coordinate near the term's female-counterpart.
  - `king` → `queen`
  - `uncle` → `aunt`
- A by-product of being able to trace vectors of meaning from one word to another → **word-vector arithmetic**.
- Canonical example: if we begin at v_{king}, subtract the vector v_{man} from it, and add the vector v_{woman} → we should find a location near the vector representing `queen`.
- Let's take a look at this arithmetic dimension by dimension!

Word-Vector Arithmetic

$$v_{king} - v_{man} + v_{woman} = v_{queen}$$
$$v_{bezos} - v_{amazon} + v_{tesla} = v_{musk}$$
$$v_{windows} - v_{microsoft} + v_{google} = v_{android}$$

Examples of word-vector arithmetic.

Word-Vector Arithmetic

$$x_{queen} = x_{king} - x_{man} + x_{woman} = -0.9 + 1.1 - 3.2 = -3.0$$
$$y_{queen} = y_{king} - y_{man} + y_{woman} = 1.9 - 2.4 + 2.5 = 2.0$$
$$z_{queen} = z_{king} - z_{man} + z_{woman} = 2.2 - 3.0 + 2.6 = 1.8$$

Word-Vector Arithmetic

- We then expect v_{queen} to be near *[-3.0, 2.0, 1.8]*.

word2viz

- Let's develop our intuitive appreciation of word vectors:
  - https://lamyiowce.github.io/word2viz/
- In the "Gender analogies" context, try adding in *pairs* of new words.
- Let's add pairs of corresponding gender-specific words like `princess` and `prince`, `duchess` and `duke`, `businesswoman` and `businessman`.
- **They fall into instructive locations**

word2viz

- The developer of word2viz, Julia Bazinska, compressed a 50-dimensional word-vector space down to **two dimensions**.
  - In order to visualize the vectors on an *xy*-coordinate system.
- This was trained on a natural language dataset consisting of 6 billion instances of 400,000 unique words.
- *x*-axis: from the words `she` to `he` as a reference point for gender.
- *y*-axis: was set to vary from a commonfolk base toward royal peak (`woman` to `queen`).

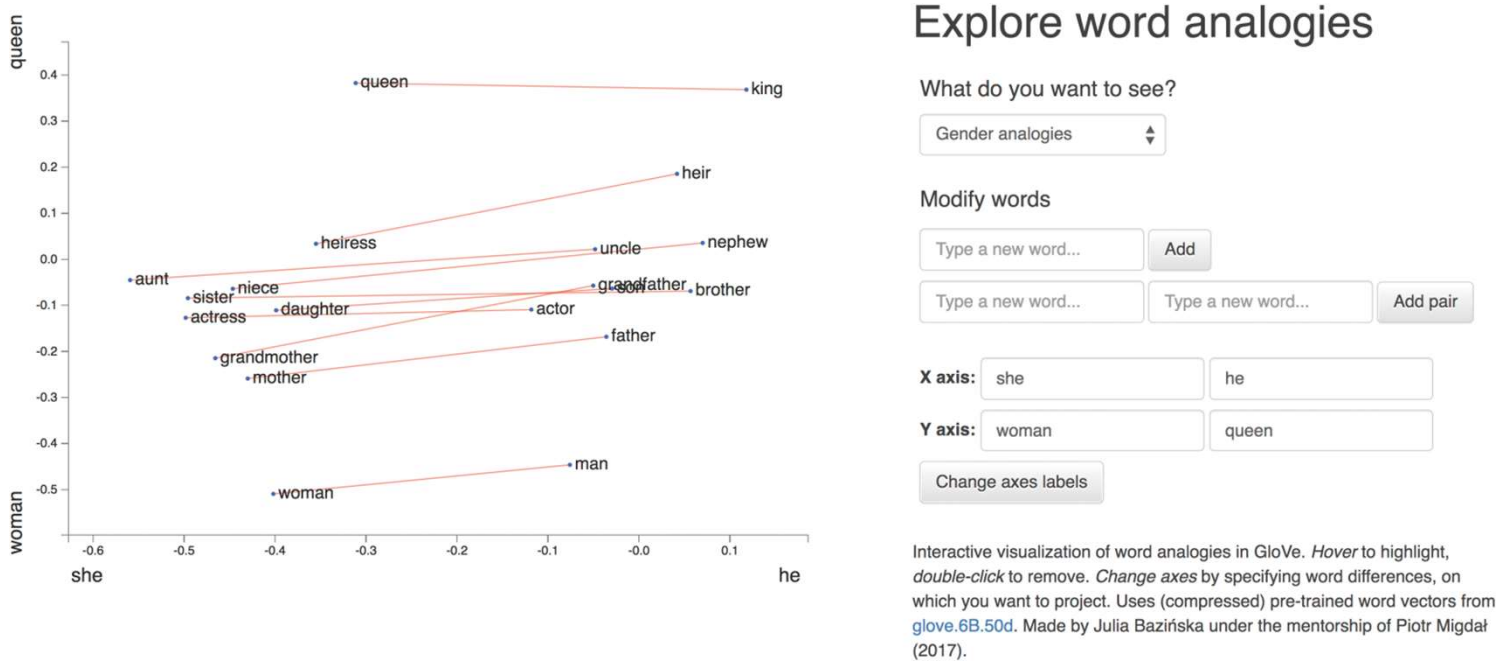
word2viz

- After this, let's experiment with other perspectives of the word-vector space.
- Let's select "Adjectives analogies" in the dropdown box.
  - Adding the words `small` and `smallest`.
- Let's change the *x*-axis labels to `nice` and `nicer`.
- And then again to `small` and `big`.

word2viz

- There's also the "Numbers say-write analogies".
- One shall change the *x*-axis to 3 and 7.
- You may build your own word2viz plot from scratch by moving to the "Empty" view.

word2viz



The default screen for word2viz, a tool for exploring word vectors interactively.

Localist Versus Distributed Representations

- We can now contrast word vectors with one-hot representations.
- **Word vectors store the meaning of words in *distributed* representation across n-dimensional space.**
  - It is distributed gradually as we move from location to location through vector space.
- **One-hot representations are *localist*.**
  - They store information on a given word within a single row of a typycally extremely sparse matrix (discrete).

Localist Versus Distributed Representations

- This table characterize more thoroughly between the one-hot approach and the vector-based to word representation.

One-Hot	Vector-Based
Not subtle	Very nuanced
Manual taxonomies	Automatic
Handles new words poorly	Seamlessly incorporates new words
Subjective	Driven by natural language data
Word similarity not represented	Word similarity = proximity in space

Localist Versus Distributed Representations

- **One-hot representations** lack nuance:
  - Simple binary flags.
- **Vector-based representations** are nuanced:
  - Information is smeared throughout a continuous space.
  - Infinity possibilities for capturing the relationship between words.
- **One-hot representations** → often require labor-intensive, manually curated taxonomies.
  - Dictionaries and language datasets.
- **Vector-based representations** → fully automatic with natural language data alone.

Localist Versus Distributed Representations

- **One-hot representations** → new row in the matrix and then reanalysis.
  - Followed by code changes.

- **Vector-based representations** → incorporated by training the vector space on corpora that includes examples of new words in their natural context.
  - New *n*-dimensional vector.
  - Positioning of all existing words remain intact.
  - Over time, as the instances of the new word increases, the accuracy of its vector-space coordinates will improve.

- **One-hot representations** → often involve **subjective interpretations** of the meaning of language.
  - They require coded rules or reference databases that are designed by developers.

- **Vector-based representations** → the meaning of language is **data driven**.

Localist Versus Distributed Representations

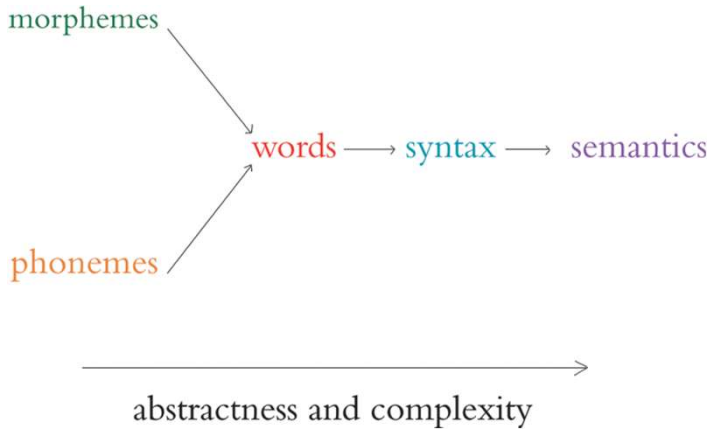
- **One-hot representations** → natively **ignore word similarity**.
  - `Couch` and `sofa` are represented no differently than unrelated words, such as `couch` and `cat`.

- In contrast, **vector-based representations** innately **handle word similarity**.

Elements of Natural Human Language

- We have considered only one element of natural human language: *word*.
- Words are made up of **language elements**.
  - Words themselves are the constituents of more complex language elements.
- We begin with language elements that make up words and build up from there.
- We will see how it is typically encoded from the traditional machine as well as **deep learning perspective**.
- **Fluid and flexible vectors X local and rigid**

Elements of Natural Human Language



Relationships between the elements of natural human language.

Elements of Natural Human Language

Representation	Traditional ML	Deep Learning	Audio-Only
Phonology	All phonemes	Vectors	True
Morphology	All morphemes	Vectors	False
Words	One-hot encoding	Vectors	False
Syntax	Phrase rules	Vectors	False
Semantics	Lambda calculus	Vectors	False

Elements of Natural Human Language

- **Phonology**: concerned with the way language sounds when it is spoken.
- Every language has a set of **phonemes**.
- Traditional ML approach → encodes segments of auditory input as specific phonemes from the language's available phonemes.
- Deep learning → model is trained to **predict phonemes** from features learned from auditory input and **represent** them in a **vector space**.
- In our classes, we work with natural language in text format.
  - Same techniques can be applied to speech data.

Elements of Natural Human Language

- **Morphology**: concerned with the forms of words.
- Every language has a set of **morphemes.**
  - Example: three morphemes `out`, `go`, and `ing` combine to form the word `outgoing`.
- Traditional ML approach → identify morphemes in text from list of al morphemes in a language.
- Deep learning → model is trained to **predict ocurrence of particular morphemes**. from features learned from auditory input and **represent** them in a **vector space**.
- Hierarchically deeper layers of neurons can combine multiple vectors into a single vector representing a word.

Elements of Natural Human Language

- Phonemes and morphemes combine to form **words**.
- We will work later with natural language data at the word level.
- Four reasons:
  - Straightforward to define what a word is.
  - It's easy to break up natural language into words via **tokenization**.
  - Most-studied level of natural language → cutting-edge techniques.
  - **Word vectors simply work well:** functional, efficient, and accurate.

Elements of Natural Human Language

- Words are combined to generate **syntax**.
- Syntax and morphology constitute the entirety of a language's grammar.
- Syntax is the arrengement of words into phrases and phrases into sentences in order to convey meaning.
- Traditional ML approach → phrases are bucketed into formal (discrete) linguistic categories.
- Deep learning → we employ vectors.
  - Every word is represented by a vector in a *n*-dimensional space.
  - Layers of neurons combine into phrases.

Elements of Natural Human Language

- Semantics is the most abstract of the elements of natural language.
  - It is concerned with the **meaning** of sentences.
- Meaning is inferred from all the underlying language elements like words and phrases (and context).
- **Complex task:** whether a passage is supposed to be taken literally or as a sacarsm can depend on subtle contextual differences (and shifting cultural norms).
- Traditional ML approach → limited in capturing semantic meaning (it doesn't the fuzziness of language).

Elements of Natural Human Language

- Deep learning → vectors come to the rescue once again.
- Representation of every **word**, every **phrase** and every **logical expression**.
- Same rationale: layers of artificial neurons can **recombine** vectors of constituent elements to **calculate semantic vectors** via the **nonlinear combination of phrase vectors**.

Google Duplex

- One of the more attention-grabbing examples of deep-learning-based NLP in recent years is that of the **Google Duplex** technology,
- Google's CEO, Sundar Pichai, held spectators in rapture as he demonstrated Google Assistant making a phone call to a Chinese food restaurant to book a reservation.
- Duplex had mastered the cadence of a human conversation, even repleting with "uh's" and "hhhm's" that we sprinkle into conversations while thinking.
- The phone call was of average audio quality and the human had a strong accent.
  - Duplex managed to make the booking.

Google Duplex

- What was impressive was the breadth of deep learning applications that had to come together to facilitate this technology.
- Consider the flow of information:
  - Sophisticated speech recognition algorithm that can process audio in real time;
  - Handle an extensive range of accents and call qualities;
  - Overcome background noise
- Human's speech faithfully transcribed + NLP model processes the sentence + decide what it means.
- Person doesn't know they're speaking to a computer and doesn't need to modulate their speech.

Google Duplex

- This means that humans responds with complex (tricky) sentences:
- *"We don't have anything tomorrow, but we have the next day and Thursday, anytime before eight. Wait no . . . Thursday at seven is out. But we can do it after eight?"*
- This is a typical poorly structured sentence that appears in natural conversarion.
- After transcription and the meaning of the sentence processed, **Duplex's NLP model conjures up a response**.
- It asks for more information if the human was unclear.
  - Otherwise it should confirme the booking.

Google Duplex

- It generates a response in text form → a text-to-speech (TTS) engine is required to synthesize the sound.
- This is a typical poorly structured sentence that only appears in natural conversarion: corrections and replacements that happen on-the-fly.
  - Duplex needs to be able to follow along.
- Speech recognition system, NLP models, and TTS engine all work in concert from the instant the call is answered.
- **Governing all of this interaction is a deep neural network that is specialized in handling information that occurs in a sequence: Recurrent Neural Network.**
  - It tracks the conversation and feeds the various inputs and outputs into the appropriate models.

Google Duplex

- In summary: Google Duplex is a sophisticated system of deep earning models that work in harmony to produce a seamless interaction on the phone.
- For now, Duplex is nevertheless limited to a few specific domains: scheduling appointments and reservations.
- It cannot carry out general conversations.
- **Even though Duplex represents a significant step forward for artificial intelligence, there is still much work to be done.**

Summary

- We learned about applications of deep learning to NLP.
- We saw the capacity for deep learning models to **automatically extract** the most pertinent **features** from data.
  - **Removing the need for labor-intensive one-hot representations of language**.
- NLP applications involving deep learning make use of vector-space embeddings.
  - **It captures the meaning of words in a nuanced manner that improves both model performance and accuracy.**

Summary

- We wil later in this course construct an NLP application by making use of neural networks:
  - That handles the input of natural language data;
  - Outputs an inference about those data.

- **Initial layers create word vectors** about those data that flow into deeper, **specialized layers** of artificial neurons.
  - Including layers that incorporate “memory”.

Referências

1. Capítulo 2 do livro *“Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence”* de Jon Krohn, Grant Beyleveld e Aglaé Bassens.