

Natural Language Processing with Machine Learning

Yogesh Haribhau Kulkarni

The Giants are at it ...

Overview of NLP with Embedding

Use case: Bank Call Center

Calling the Call Center

- Calling to an IVR (Integrated voice response)
- A prerecorded menu selection.
- “Please press 1 for Account Details, Please press 2 for ...”
- till it comes to your option.
- towards end, somewhere, given access to a person to talk to.

Boring? Annoying?



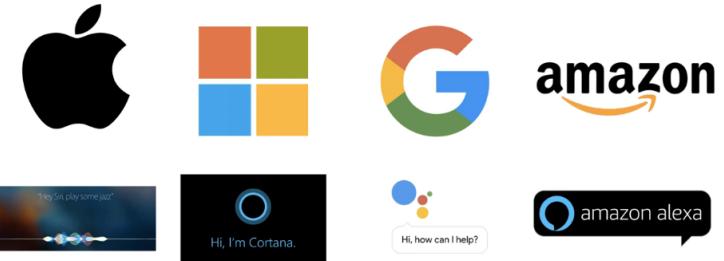
(Ref: Deep Learning and NLP A-Z - Kirill Eremenko)

Instead, how about typing/saying your query directly and getting the answer right away?

Solution

Chatbots

- Which problem of IVR it is solving?
- Advantages?
- Disadvantages?
- Gaining popularity ...
- Many platforms
- Companies in Pune?



(Ref: Deep Learning and NLP A-Z - Kirill Eremenko)

- Chatbots or QA systems, predominantly voice based,
- Underlying processing is primarily Natural Language Processing (NLP).
- You can have your own chatbot, specific to you!!
- NLP is the core skill needed.

Why so much popularity?

Chatbots are:

- Autonomous and Always Available
- Drive Conversation
- Able to handle millions of requests, scalable.

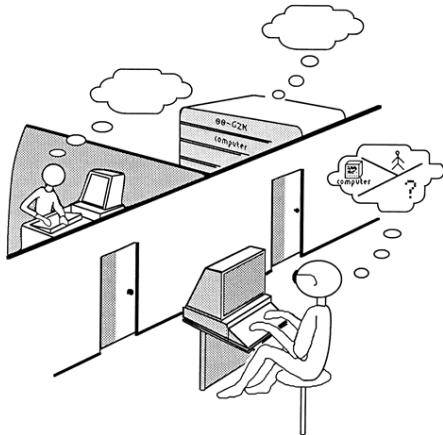
Its hard to master language, and thus NLP.

NLP is AI-complete

- “The most difficult problems in AI manifest themselves in human language phenomena.”
- Use of language is the touchstone of intelligent behavior.
- Test for Intelligence - Turing Test
- Alan Turing (1950) proposed a test of a machine’s capability to perform human-like conversation.

Turing Test

A human judge engages in a natural language conversation with two other parties, one a human and the other a machine; if the judge cannot reliably tell which is which, then the machine is said to pass the test.



Early Conversational Programs

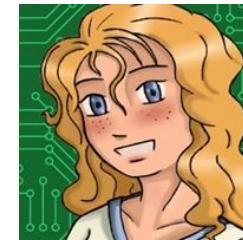
- ELIZA (by Joseph Weizenbaum), 1966.
 - A psychotherapist, but NO real understanding;
 - Simple pattern-matching to respond to user input to canned responses

(my ?x depresses me) (why does your ?x depress you)
(life ?x) (why do you say it ?x)
(I could ?x) (you could ?x)
(because ?x) (that is a good reason)
(?x) (tell me more)

```
>> Hello.  
How do you do. Please state your problem.  
>> Men are all alike.  
In what way ?  
>> They're always bugging us about something or other.  
I'm not sure I understand you fully.  
>> They're always bugging us about something or other.  
Can you think of a specific example ?  
>> Well, my boyfriend made me come here.  
Your boyfriend made you come here ?  
>> He says I'm depressed much of the time.  
I am sorry to hear that you are depressed.
```

Loebner Prize

- In 1990, Hugh Loebner started Turing Test competition
 - \$100,000 will be awarded to the first bot that judges cannot distinguish from a real human in a Turing test that includes text, visual, and auditory input.
 - Nobody has won the grand prize yet.
 - 2016 (and 2013) year-wise top winner - Mitsuku. <https://www.facebook.com/mitsukubot>



Why can't we win the Grand Prize? What are the challenges? Why Language is hard? What is Language?

What is Language?

Language Types

Natural Language

日本語で
かゆ せかいかくち
冬は世界各地でさまざまなお祝いが行わられる時期で
ほんのいくつか例を挙げるだけでも、ハナカ、クリスマス、マスクランザ、新年などさまざまなお祝いがあります。
かぶんかぶん
各文化によってその扱い方はましまさまですが、ほとん
かゆ いわゆるお祝いにはごちそうが欠かせません。

... waren
Botschaften zu besprechen und Gesprächspartner auf uns bestimmt. Andererseits sah man sich unmittelbar zu den Anreisen der Botschaften und Operatoren auf uns und bestimmt. Andererseits sah man sich unmittelbar zu den Anreisen der Botschaften und Operatoren auf uns bestimmt. Andererseits sah man sich unmittelbar zu den Anreisen der Botschaften und Operatoren auf uns bestimmt.

(http://www.transparent.com/learn-japanese/articles/dec_99.html)

Artificial Language

```
try {
    cMessage = messageQueue.take();
    for (AsyncContext ac : queue) {
        try {
            PrintWriter acWriter = ac.getWriter();
            acWriter.println(cMessage);
            acWriter.flush();
        } catch (IOException e) {
            System.out.append((char) e);
            append(CharArray queue);
        }
    }
} catch (InterruptedException e) {
    System.out.format(Locale.US, "%s", e);
}
```

```

def add(x,y):
    return x+y

def dotester(ast):
    if len(ast) >= 2:
        if ast[0] == 'node':
            if len(ast) >= 2:
                if ast[1] == 'name':
                    label = ast[1]
                    if len(ast) >= 2:
                        if ast[2] == 'kids':
                            kids = ast[3]
                            if len(kids) >= 1:
                                for kid in kids:
                                    if isinstance(kid, str):
                                        print("  " + kid)
                                    else:
                                        print("  " + kid[1])
                else:
                    print("  " + ast[1])
            else:
                print("  " + ast[1])
        else:
            print("  " + ast[0])
    else:
        print("  " + ast[0])
    children = []
    for n, child in enumerate(ast[1:]):
        children.append(dotester(child))
    print("  " + ast[0] + " has " + str(len(children)) + " children")
    for name in children:
        print("    " + name[1] + " is named " + name[0])

```

Differences?

Language, simplistically

- A vocabulary consists of a set of words
 - A text is composed of a sequence of words from a vocabulary
 - A language is constructed of a set of all possible texts



(<http://learnenglish.britishcouncil.org/en/vocabulary-games>)

THIS WEEK

Beyond the genome

Silencing the expression of many healthy and disease-associated genes could provide a new way to treat genetic diseases.

The genome is the complete set of genetic material in a cell. It contains all the information needed to build and maintain an organism, passing it on to the next generation, and enabling the growth, development, reproduction and survival of the organism. The genome is composed of DNA, which is organized into chromosomes.

Genes are segments of DNA that code for specific proteins or functional RNA molecules. They are the basic units of heredity and are passed from parents to offspring. Genes are located on chromosomes, which are made of DNA and protein. Chromosomes are found in the nucleus of cells.

Chromosomes are composed of DNA and protein. They are found in the nucleus of cells. Chromosomes are composed of DNA and protein. They are found in the nucleus of cells.

(http://www.nature.com/polopoly_fs/1.16929/menu/main/topColumns/topLeftColumn/pdf/518273a.pdf)



(<http://www.old-englisch.de/language.php>)

NLP

- NLP is Natural Language Processing, ie processing Natural Langauge for some end-purpose in mind.
- Inspite of usage of Natural Language for thousands of years, why are we not able to process it well?

NLP Challenges

Paraphrasing

Paraphrasing: Different words/sentences express the same meaning

- Season of the year: Fall/Autumn
- Book delivery time
 - When will my book arrive?
 - When will I receive my book?

Ambiguity

Ambiguity: One word/sentence can have different meanings

- Fall
 - The third season of the year
 - Moving down towards the ground or towards a lower position
- The door is open
 - Expressing a fact
 - A request to close the door

Syntax and ambiguity

"I saw the man with a telescope." - Who had the telescope?

Semantics

The astronomer loves the star.

- Star in the sky
- Celebrity



(<http://en.wikipedia.org/wiki/Star#/media/File:Starsinthesky.jpg>)



(<http://www.businessnewsdaily.com/2023-celebrity-hiring.html>)

NLP Applications

Grammar

Spell and Grammar Checking

- Checking spelling and grammar
- Suggesting alternatives for the errors

Google narural language processing

All Images News Videos Books More ▾ Search tools

About 28.500.000 results (0.45 seconds)

Showing results for **natural language processing**
Search instead for **narural language processing**

Word Prediction

Word Prediction: Predicting the next word that is highly probable to be typed by the user

- Mobile typing
- Search Engines

File Edit View History Bookmarks Tools Help

Google https://www.google.de/gfe_rd=cr&ei=cJLU-SZEMqEtQa89YDgBA

Google natural language processing

natural language processing
natural language
natural language system
natural language processing with python

Weitere Informationen

Information Retrieval

Information Retrieval: Finding relevant information to the user's query

A screenshot of a Google search results page for the query "panama papers". The results include links to "Datenleak Panama Papers - sueddeutsche.de", "The Panama Papers - ICIJ", and "Panama Papers - Wikipedia". Below the main results, there is a section titled "In the news" with links to BBC News articles about Putin rejecting corruption allegations and David Cameron admitting to profiting from the Panama Papers.

Text Categorization

Text Categorization: Assigning one (or more) pre-defined category to a text

A screenshot of a PubMed search results page for the paper "Coupling of angiogenesis and osteogenesis by a specific vessel subtype in bone". The page shows the abstract, author information, and MeSH terms related to the paper, such as "Angiogenesis", "Osteogenesis", and "Blood Vessels".

Text Categorization



Classify

Classify method: text url

Enter url to download and classify with:

<http://edition.cnn.com/2015/02/18/football/cl>

uClassify!

Remove html

1. Sports (92.8 %)
2. Entertainment (4.8 %)
3. Men (0.7 %)

[Show all classifications >>](#)

Summarization

Summarization: Generating a short summary from one or more documents, sometimes based on a given query



This is a [7] sentence summary of <http://hpi.de/en/news/jahrgaenge/2015/des...>

Summary processing at low priority, upgrade to **BOOST**

Design Thinking Week: Students Improve the Daily Life Experience for People with Illiteracies

On the occasion of the World Literacy Day on September 8 more than 40 young innovators applied their Design Thinking skills in order to make life easier for these people.

Here, the focus was especially on the possibilities of using digital technologies and computers to better the daily obstacles in life of the people concerned.

Under the guidance of the D-School's coaches the teams researched, developed and prototyped - and could present many versatile solutions in the end: e.g. one of the groups came up with an idea for a software program that lets internet browsers read texts, functions and links out loud so that people with reading problems can still use news sites or social networks like Facebook.

<http://smmry.com/>

Question answering

Question answering: Answering questions with a short answer

START
Natural Language Question Answering System

what countries speak Spanish Ask Question >

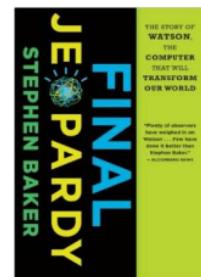
==> what countries speak Spanish

The language Spanish is spoken in Argentina, Aruba, Belize, Bolivia, Brazil, Canada, Cayman Islands, Chile, Colombia, Costa Rica, Cuba, Curacao, Dominican Republic, Ecuador, El Salvador, Equatorial Guinea, Falkland Islands (Islas Malvinas), Gibraltar, Guatemala, Honduras, Mexico, Nicaragua, Panama, Paraguay, Peru, Puerto Rico, Saint Martin, Sint Maarten, Spain, Switzerland, Trinidad and Tobago, United States, Uruguay, Venezuela, and Virgin Islands.

The language Castilian Spanish is spoken in Spain.

Question answering

Question answering: IBM Watson in Jeopardy



Information Extraction

Information Extraction: Extracting important concepts from texts and assigning them to slots in a certain template



Merkel at the EPP Summit, March 2016

Chancellor of Germany
Incumbent
Assumed office
22 November 2005
President
Horst Köhler Christian Wulff Joachim Gauck
Deputy
Franz Müntefering Frank-Walter Steinmeier Guido Westerwelle Philipp Rösler Sigmar Gabriel
Preceded by
Gerhard Schröder
Leader of the Christian Democratic Union
Incumbent
Assumed office

In office
17 November 1990 – 26 October 1998
Chancellor
Helmut Kohl
Preceded by
Klaus Töpfer
Succeeded by
Jürgen Trittin
Minister for Women and Youth
In office
18 January 1991 – 17 November 1994
Chancellor
Helmut Kohl
Preceded by
Ursula Lehr
Succeeded by
Claudia Nette
Personal details
Born
Angela Dorothea Kasner 17 July 1954 (age 61) Hamburg, West Germany
Political party
Democratic Awakening (1989–1990) Christian Democratic Union (1990–present)
Spouse(s)
Ulrich Merkel (1977–1982) Joachim Sauer (1998–present)
Alma mater
Leipzig University
Religion
Lutheranism (within Evangelical Church)
Signature

Information Extraction

Information Extraction: Includes named-entity recognition

lancet
a Medication Event Extraction System for Clinical Text

Project Home Downloads Wiki Issues Source Summary People

Project Information

Started by 1 user Project feeds

Code license GNU GPL v2

Labels medication, extractor, lancet, discharge, summary, (db2, NLP, challenge, 2009)

Members lizuof...@gmail.com

Lancet is a supervised machine-learning system that automatically extracts medication events consisting of medication names and information pertaining to their prescribed use (dosage, mode, frequency, duration and reason) from lists or narrative text in medical discharge summaries.

Thus, she was transitioned over to a ciprofloxacin 700 mg po b.i.d. regime for a total of 12 days for a presumed urinary tract infection. [native]

■ = medication ■ = dosage ■ = manner ■ = frequency ■ = duration ■ = reason

Machine Translation

Machine Translation: Translating a text from one language to another

Google

Translate

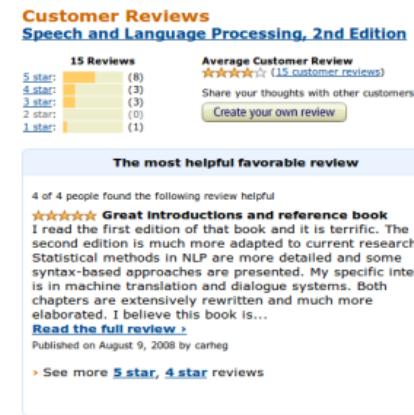
German Portuguese Spanish Detect language English Portuguese German Translate

Die Lehre am Hasso-Plattner-Institut richtet sich an begabte junge Leute, die praxisnah zu IT-Ingenieuren ausgebildet werden wollen.

Ensinar no Instituto Hasso Plattner é destinado a jovens talentosos que querem ser treinados para a prática de engenheiros de TI.

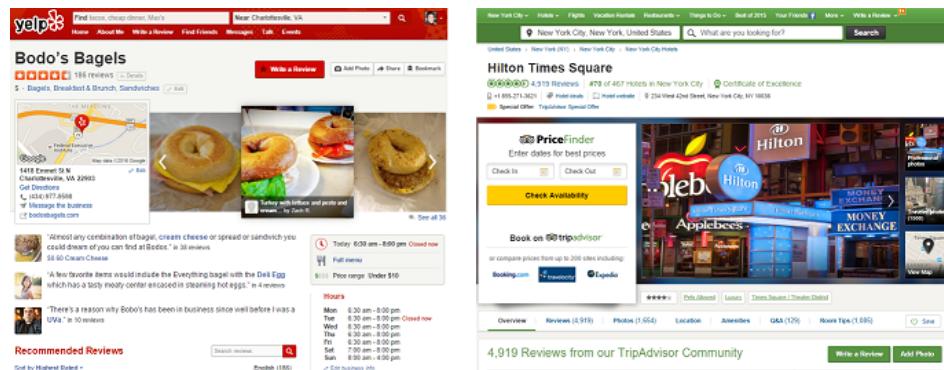
Sentiment Analysis

Sentiment Analysis: Identifying sentiments and opinions stated in a text



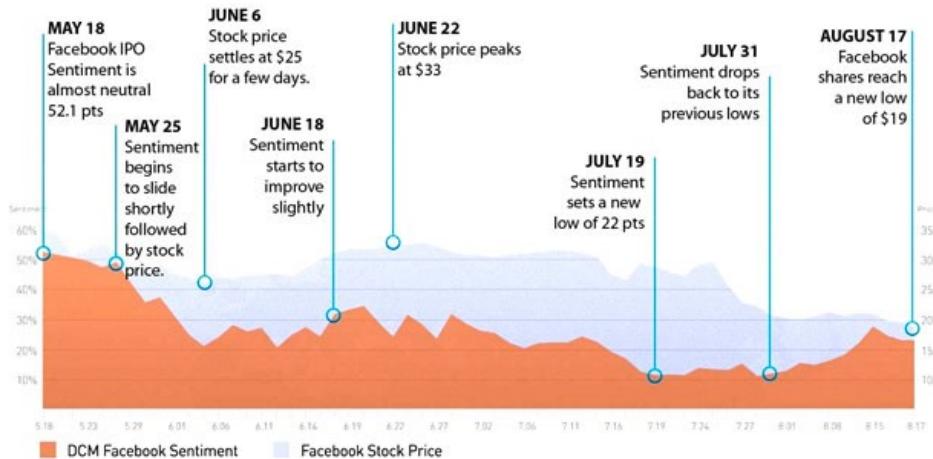
Sentiment Analysis

Restaurant/hotel recommendation, Product reviews



Sentiment Analysis

Text analytics in financial services



NLP in today's time

Trends:

- An enormous amount of information is now available in machine readable form as natural language text (newspapers, web pages, medical records, financial filings, product reviews, discussion forums, etc.)
- Conversational agents are becoming an important form of human-computer communication
- Much of human-human interaction is now mediated by computers via social media

Collectively, this means that copious data is available to be used in the development of NLP systems.

Level of difficulties

- Easy (mostly solved)
 - Spell and grammar checking
 - Some text categorization tasks
 - Some named-entity recognition tasks
- Intermediate (good progress)
 - Information retrieval
 - Sentiment analysis
 - Machine translation
 - Information extraction
- Difficult (still hard)
 - Question answering
 - Summarization
 - Dialog systems

Language Representation: How to make text computable?

Document Representation & Language Model

- How to represent a document?
- Make it computable
- How to infer the relationship among documents or identify the structure within a document?
- Knowledge discovery
- Language Model and N-Grams

Bag-of-Words representation

Term as the basis for vector space

- Doc1: Text mining is to identify useful information.
- Doc2: Useful information is mined from text.
- Doc3: Apple is delicious.

	text	information	identify	mining	mined	is	useful	to	from	apple	delicious
Doc1	1	1	1	1	0	1	1	1	0	0	0
Doc2	1	1	0	0	1	1	1	0	1	0	0
Doc3	0	0	0	0	0	1	0	0	0	1	1

What are Word Vectors/Embeddings?

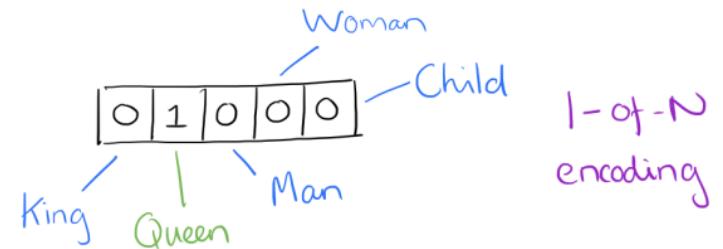
- Word Embeddings are the texts converted into numbers
- There may be different numerical representations of same text.
- Many Machine Learning algorithms and almost all Deep Learning Architectures are incapable of processing strings or plain text in their raw form.
- They require numbers as inputs to perform any sort of job, be it classification, regression etc. in broad terms.
- So, for the computer to be able to "understand" a vector representation of a word is required.

Different types of Word Vectors

- (Traditional) Frequency based Embedding:
 - One-hot
 - Count Vector
 - TF-IDF Vector
 - Co-Occurrence Vector
- (Modern) Prediction based Embedding:
 - Word2vec (Google)
 - Global Vector Representations (GloVe) (Stanford)

One Hot

One-hot: Suppose our vocabulary has only five words: King, Queen, Man, Woman, and Child. We could encode the word 'Queen' as:



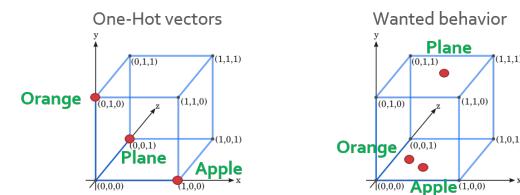
No meaningful comparison possible.

Good Vector Representation

- To have "Semantic" (meaning-wise) representation, the Similar words should be close to each other in the hyper dimensional space.
- Non-similar words should be far apart from each other in the hyper dimensional space.

Good Vector Representation

- Traditional One Hot Encoding:
 - Apple = [1, 0, 0]
 - Orange = [0, 1, 0]
 - Plane = [0, 0, 1]



- Very few cells participate in the representation.

Word2Vec

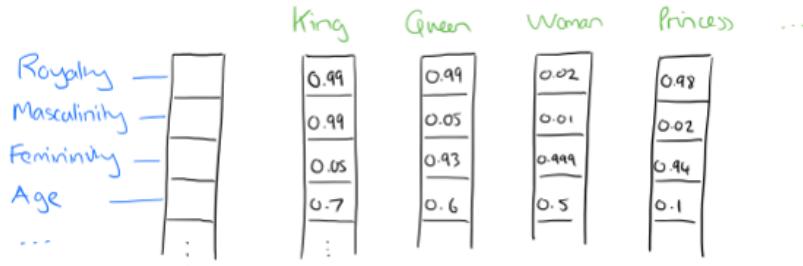
Word2vec (Google): a distributed representation of a word is used and not sparse like One-Hot.



Represent in some abstract way the ‘meaning’ of a word.

Word Distributed Representation - Word2Vec

- All vector cells participate in representing each word.
- Words are represented by real valued dense vectors of significantly smaller dimensions (e.g. 100 - 1000).
- Intuition: consider each vector cell as a representative of some feature.



Word Representations Comparison

Traditional Method - Bag of Words Model

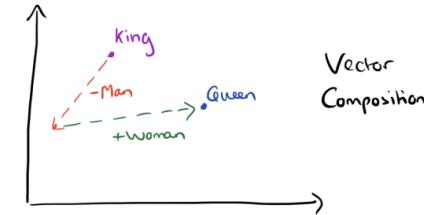
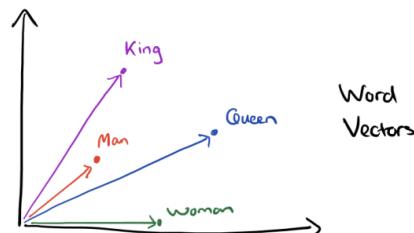
- Uses one hot encoding
- Each word in the vocabulary is represented by one bit position in a **HUGE** vector.
- For example, with a vocabulary of 10000 words, and "Hello" is the 4th word in the dictionary: 0 0 0 1 0 0 0 0 0 0
- Context information is not utilized

Modern - Word Vectors

- Stores each word in as a point in space, represented by a vector of fixed number of dimensions (generally 300)
- Unsupervised, built just by reading huge corpus
- For example, "Hello" might be represented as : [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02]
- Context information is utilized

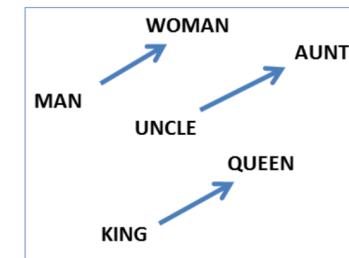
Examples

Vectors for King, Man, Queen, & Woman:

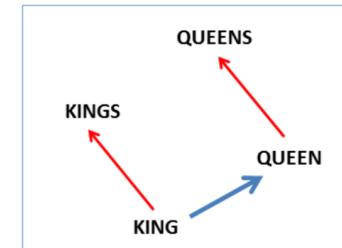


Examples

Gender relation:



Plural relation:



Examples

Word pair relationships:

Table 8: Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).

Relationship	Example 1	Example 2	Example 3
France - Paris big - bigger	Italy: Rome small: larger	Japan: Tokyo cold: colder	Florida: Tallahassee quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France copper - Cu	Berlusconi: Italy zinc: Zn	Merkel: Germany gold: Au	Koizumi: Japan uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Examples

Country-capital city relationship:

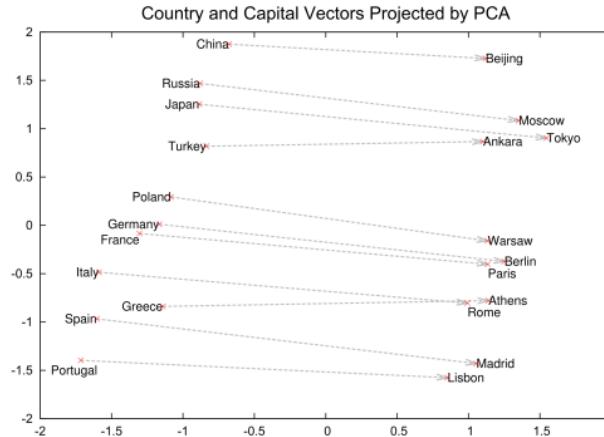


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

The Power of Word2Vecs

- They provide a fresh perspective to ALL problems in NLP, and not just solve one problem.
 - Technological Improvement
 - Rise of deep learning since 2006 (Big Data + GPUs + Work done by Andrew Ng, Yoshua Bengio, Yann Lecun and Geoff Hinton)
 - Application of Deep Learning to NLP - led by Yoshua Bengio, Christopher Manning, Richard Socher, Tomas Mikalov
 - The need for unsupervised learning . (Supervised learning tends to be excessively dependent on hand-labeled data and often does not scale)

NLP Activities: How to process text?

Document/Section splitting

Document/Section splitting: Splitting a text into sections

Environ Biol Fish (2012) 95:103–114
DOI 10.1007/s10641-012-0034-x

regression modeling techniques to identify the most significant and independent 3D image findings. A P value

Sentence splitting

Sentence splitting: Splitting a text into sentences

11 Sentences (= "T-" or "Terminable" units be want")

Average 23.55 words (SD=12.10)

OBJECTIVES: To investigate the correlation of three-dimensional (3D) ultrasound features with prognostic factors in invasive ductal carcinoma.

METHODS: Surgical resection specimens of 85 invasive ductal carcinomas of 85 women who had undergone 3D ultrasound were included.

Morphology features and vascularization perfusion on 3D ultrasound were evaluated. Pathologic prognostic factors, including tumour size, histological grade, lymph node status, oestrogen and progesterone receptor status (ER,

Pathologic prognostic factors, including tumour size, histological grade, lymph node status, oestrogen receptor (ER), progesterone receptor (PR), c erbB-2 and p53 expression, and microvessel density (MVD) were determined.

RESULTS: The regression pattern in the coronal plane has a significant value as an independent predictor of a small tumour size ($P = 0.02$), a lower histological grade ($P = 0.02$), and positivity ER or PR expression status ($P = 0.02$) ($P < 0.001$, $n = 0.044$).

The resection pattern with a tumor capsule did not relate to low grade and ER-positive tumors.

The retraction pattern with a hyperechoic ring only existed in low-grade and ER-positive tumours.

The increased intra-tumour vascularization index (VI, the mean tumour vascularity) reflected a higher histological grade ($P = 0.025$) and had a significant correlation with the MFI ($P = 0.001$) ($n = 950$, $P < 0.001$, $r = 0.024$).

CONCLUSIONS: The retraction pattern and histogram indices of VI provided by 3D ultrasound may be useful in predicting prognostic information about breast cancer.

KEY POINTS: • Three-dimensional ultrasound can potentially provide prognostic evaluation of breast cancer. • The retraction pattern and hypercholeic ring in the coronal plane suggest good prognosis. • The increased intra-tumour vascularization index reflects a higher histological grade. • The intra-tumour vascularization index is positively correlated with microvesSEL density.

Tokenization

Tokenization

- Process of breaking a stream of text up into tokens (= words, phrases, symbols, or other meaningful elements)
 - Typically performed at the “word” level
 - Not easy: Hewlett-Packard, U.S.A., in some languages there is no “space” between words!

Stemming

Stemming

- Reduces similar words to a given “stem”
- E.g. detects, detected, detecting, detect : detect (stem).
- Usually set of rules for suffix stripping
- Most popular for English: Porter’s Algorithm
- 36% reduction in indexing vocabulary (English)
- Linguistic correctness of resulting stems not necessary (sensitivities : sensit)

Lemmatization

Lemmatization

- Uses a vocabulary and full morphological analysis of words
- Aims to remove inflectional endings only
- Return the base or dictionary form of a word, which is known as the lemma.
- E.g. saw : see, been, was : be

Part-of-speech tagging

Part-of-speech tagging: Assigning a syntactic tag to each word in a sentence

Stanford Parser

Please enter a sentence to be parsed:

Surgical resection specimens of 85 invasive ductal carcinomas of 85 women who had undergone 3D ultrasound were included.

Language: Sample Sentence

Your query

Surgical resection specimens of 85 invasive ductal carcinomas of 85 women who had undergone 3D ultrasound were included.

Tagging

Surgical/NNP resection/NN specimens/NNS of/IN 85/CD invasive/JJ ductal/JJ carcinomas/NNS of/IN 85/CD women/NNS who/WP had/VBD undergone/VBN 3D/CD ultrasound/NN were/VBD included/VBN ./.

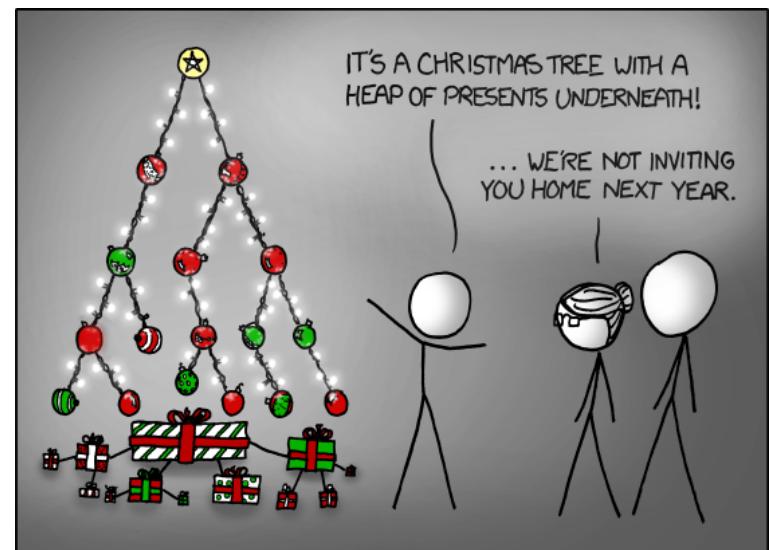
Parse

```
(ROOT
  (S
    (NP
      (NP (NP (NNP Surgical) (NN resection) (NNS specimens))
        (PP (IN of)
          (NP
            (NP (CD 85) (JJ invasive) (JJ ductal) (NNS carcinomas))
              (PP (IN of)
                (NP
                  (NP (CD 85) (NNS women))
                    (SBAR
                      (WHNP (WP who))
                        (S
                          (VP (VBD had)
                            (VP (VBN undergone)
                              (NP (CD 3D) (NN ultrasound)))))))))))
                (VP (VBD were)
                  (VP (VBN included))))
              (. .)))
```

<http://nlp.stanford.edu:8080/corenlp/>

Parsing

((DaimlerChryslershares)NP(rose(threeeights)NUMP(to22)PP-NUM)VP)s



Parsing

Parsing: Building the syntactic tree of a sentence

Syntax Tree

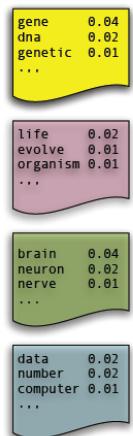
Syntax: Sample English grammar

<http://nlp.stanford.edu:8080/corenlp/>

$S \rightarrow NP VP$
 $S \rightarrow Aux NP VP$
 $S \rightarrow VP$
 $NP \rightarrow Pronoun$
 $NP \rightarrow Proper-Noun$
 $NP \rightarrow Det Nominal$
 $Nominal \rightarrow Noun$
 $Nominal \rightarrow Nominal Noun$
 $Nominal \rightarrow Nominal PP$
 $VP \rightarrow Verb$
 $VP \rightarrow Verb NP$
 $VP \rightarrow Verb NP PP$
 $VP \rightarrow Verb PP$
 $VP \rightarrow VP PP$
 $PP \rightarrow Preposition NP$

$Det \rightarrow that | this | a$
 $Noun \rightarrow book | flight | meal | money$
 $Verb \rightarrow book | include | prefer$
 $Pronoun \rightarrow I | she | me$
 $Proper-Noun \rightarrow Houston | TWA$
 $Aux \rightarrow does$
 $Preposition \rightarrow from | to | on | near | through$

Topics



Documents

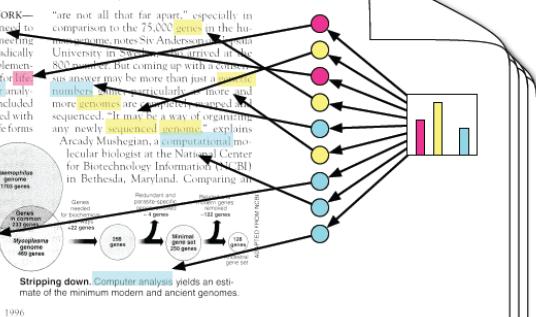
Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does a *minimal* life form need? Last week at the genome meeting here, two-genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in model organisms and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, these predictions

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

SCIENCE • VOL. 272 • 24 MAY 1996



"are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siegfried J. Stukenberg, a University of Michigan molecular biologist. But coming up with a concrete answer may be more than just a game of numbers. "It's particularly interesting and more genomes are definitely needed to be sequenced," he says. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

estimated minimum set of genes to a

known set of genes

Word embeddings

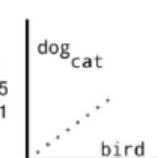
Word embeddings: Compute a vector representing the distributed representation for every word

A

$$\begin{matrix} & 1 & 2 & \dots \\ & \text{document} & \text{document} & \dots \\ \text{term 1} & & & \\ \text{term 2} & & & \\ \text{term 3} & & & \\ \vdots & & & \\ \text{term } n & & & \end{matrix} \xrightarrow{\text{term-document matrix}} \text{truncated singular value decomposition} = \boxed{1^{\text{st}} \text{ singular component}} + \boxed{2^{\text{nd}} \text{ singular component}} + \dots + \boxed{k^{\text{th}} \text{ singular component}}$$

B

$$\begin{matrix} \text{cat - dog:.50} \\ \text{cat - bird:.36} \\ \text{dog - bird:.36} \\ \vdots \end{matrix} \xrightarrow{\text{SVD truncation}} \begin{matrix} \text{cat - dog:.62} \\ \text{cat - bird:.005} \\ \text{dog - bird:.001} \\ \vdots \end{matrix}$$



Named-entity recognition

Named-entity recognition: Identifying pre-defined entity types in a sentence

The interface includes a sidebar for highlighting entity types like Anatomy, Diseases, Chemicals, Genes and Proteins, Cellular Components, Molecular Functions, Biological Processes, and Ambiguous. The main area shows a text snippet about Duchenne muscular dystrophy (DMD) and its relation to inflammatory cells. It also shows a concept tree with categories like Anatomy, Disorders, Chemicals, Genes and Proteins, Cellular Components, Molecular Functions, and Biological Processes. A status bar at the bottom indicates 46 annotated concepts.

Topic modelings

Topic modeling: Identifying structures in the text corpus

Introduction

- Natural Language Processing (NLP) and Machine Learning (ML) form the foundation for modern intelligent systems.
- Retrieval Augmented Generation (RAG) leverages NLP pipelines, embeddings, and ML for knowledge-grounded responses.
- This presentation explores:
 - Text preprocessing and representation
 - spaCy-based NLP components

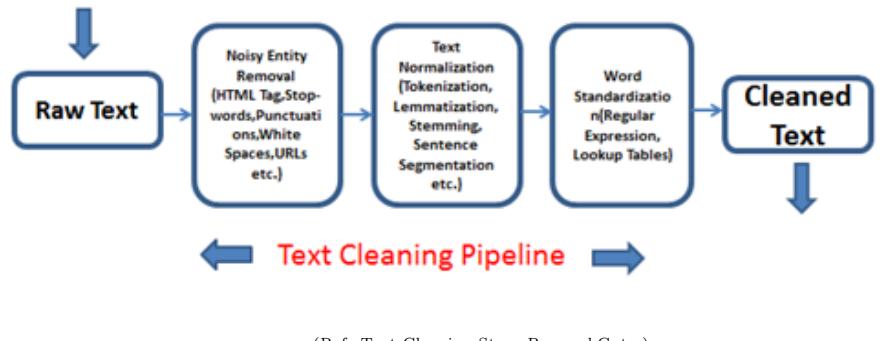
NLP with Spacy

- ML techniques for classification and clustering
- Embedding-based search and semantic understanding
- Sentiment analysis applications



Text Preprocessing

- Clean and normalize raw text data for consistent downstream analysis.
- Common steps include lowercasing, punctuation & digit removal, and whitespace normalization.
- Ensures reliable tokenization and model training.
- Helps retain semantic integrity by focusing on meaningful linguistic content.



Tokenization

- Tokenization splits text into smaller linguistic units (words, subwords, or sentences).
- spaCy provides efficient rule-based and statistical tokenizers.
- Basis for vectorization, tagging, and parsing.

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("Retrieval-Augmented Generation is powerful!")
tokens = [token.text for token in doc]
print(tokens)
# ['Retrieval', '-', 'Augmented', 'Generation', 'is', 'powerful', '!']
```

Lowercasing & Cleaning

- Lowercasing removes case sensitivity in NLP models.
- Cleaning eliminates noise: URLs, HTML tags, punctuation, numbers.
- Helps search and RAG systems achieve higher recall and consistency.

```
import re
text = "Visit https://openai.com for <b>AI Research!</b>"
cleaned = re.sub(r"http\S+|<.*?>", "", text).lower()
print(cleaned)
# 'visit for ai research!'
```

Stop Word Removal

- Removes high-frequency words that add little meaning.
- Reduces dimensionality and improves efficiency of embeddings.
- Example stop words: "the", "is", "in", "on".

```

from spacy.lang.en.stop_words import STOP_WORDS
tokens = ["Retrieval", "Augmented", "Generation", "is", "powerful"]
filtered = [w for w in tokens if w.lower() not in STOP_WORDS]
print(filtered)
# ['Retrieval', 'Augmented', 'Generation', 'powerful']

```

Stemming & Lemmatization

- Stemming removes suffixes using heuristic rules.
- Lemmatization uses grammar-based transformations to return dictionary form.
- Reduces vocabulary size and enhances text matching in retrieval.

```

from nltk.stem import WordNetLemmatizer
lem = WordNetLemmatizer()
print(lem.lemmatize("running", "v")) # 'run'
print(lem.lemmatize("better", "a")) # 'good'

```

Regular Expressions

- Enable pattern-based extraction from text.
- Examples: extract emails, phone numbers, or URLs.
- Essential for data cleaning, anonymization, and text mining.

```

import re
text = "Contact us at info@company.com or sales@domain.org"
emails = re.findall(r"\b[A-Za-z0-9._%+-]+@[A-Za-z]+\.[A-Z|a-z]{2,}\b", text)
print(emails)
# ['info@company.com', 'sales@domain.org']

```

Project: Text Preprocessing Pipeline

- Modular preprocessing pipeline for flexible text workflows.
- Each function handles one cleaning step.
- Easily extendable for lemmatization, stemming, or entity masking.

```

class TextPipeline:
    def __init__(self, steps):
        self.steps = steps # list of functions

    def run(self, text):
        for fn in self.steps:
            text = fn(text)
        return text

pipeline = TextPipeline([basic_clean])

```

```

print(pipeline.run("Hello World! This is 2025."))

```

Project: Document Cleaner

- Extracts text from PDFs, HTML, and Word documents.
- Normalizes whitespace, removes headers/footers, handles encoding.
- Outputs clean text ready for embedding or indexing.

```

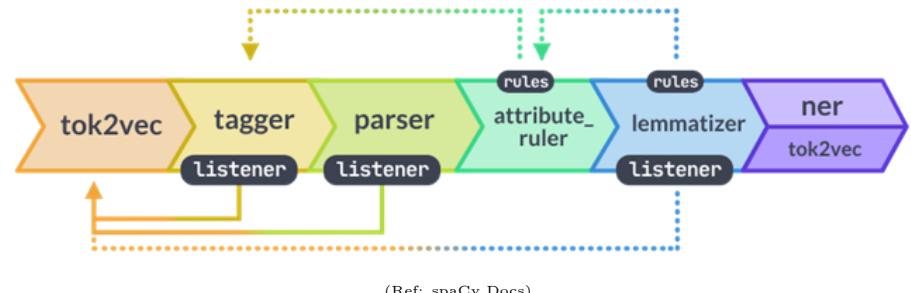
from bs4 import BeautifulSoup
from PyPDF2 import PdfReader

def clean_document(path):
    if path.endswith(".pdf"):
        text = "\n".join([page.extract_text() for page in PdfReader(path).pages])
    elif path.endswith(".html"):
        html = open(path).read()
        text = BeautifulSoup(html, "html.parser").get_text()
    else:
        text = open(path).read()
    return re.sub(r"\s+", " ", text.strip())

```

spaCy Pipeline Basics

- spaCy's pipeline includes tokenization, tagging, parsing, and entity recognition.
- Designed for efficiency and extensibility.
- Supports custom components for preprocessing and postprocessing.



Part-of-Speech Tagging

- Assigns syntactic roles: noun, verb, adjective, etc.
- Enables grammar-aware retrieval, information extraction, and text summarization.

```

doc = nlp("NLP models enhance communication.")
for token in doc:
    print(token.text, token.pos_)
# Output: NLP NOUN, models NOUN, enhance VERB, communication NOUN

```

Named Entity Recognition (NER)

- Identifies named entities: PERSON, ORG, LOC, DATE, MONEY, etc.
- Crucial for knowledge extraction and linking to external databases.

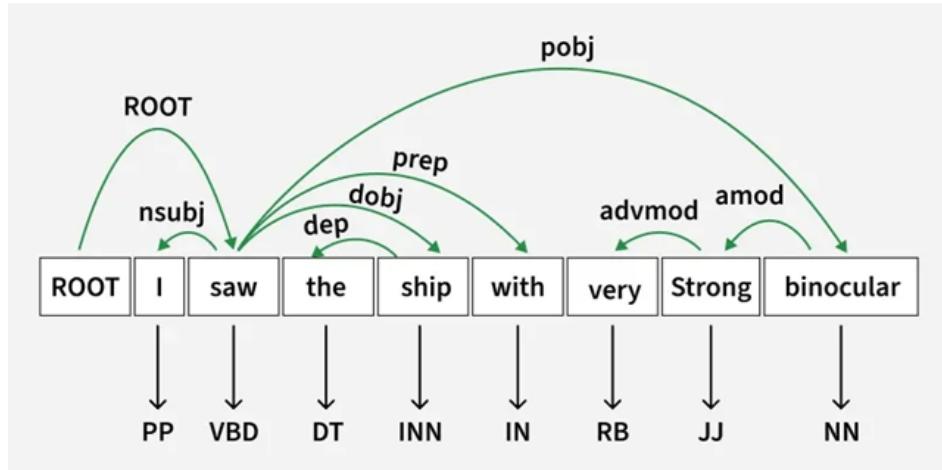
```

doc = nlp("OpenAI was founded in San Francisco in 2015.")
for ent in doc.ents:
    print(ent.text, ent.label_)
# OpenAI ORG, San Francisco GPE, 2015 DATE

```

Dependency Parsing

- Analyzes grammatical structure of sentences.
- Reveals subject-object relationships.
- Basis for relation extraction and semantic role labeling.



Text Similarity

- Measures semantic closeness between two texts.
- Useful in RAG for ranking documents by contextual relevance.

```

doc1 = nlp("AI improves healthcare")
doc2 = nlp("Artificial intelligence helps medicine")
print(doc1.similarity(doc2))
# Output: similarity score ~0.85

```

Project: Named Entity Extractor

- Extracts and exports named entities from text.
- Can be integrated with visualization tools or stored in databases.

```

def extract_entities(text):
    nlp = spacy.load("en_core_web_sm")
    doc = nlp(text)
    return {ent.text: ent.label_ for ent in doc.ents}

print(extract_entities("Elon Musk leads SpaceX and Tesla."))

```

Project: Text Similarity Engine

- Uses embeddings to compute cosine similarity between texts.
- Enables semantic document search and clustering.

```

from sklearn.metrics.pairwise import cosine_similarity

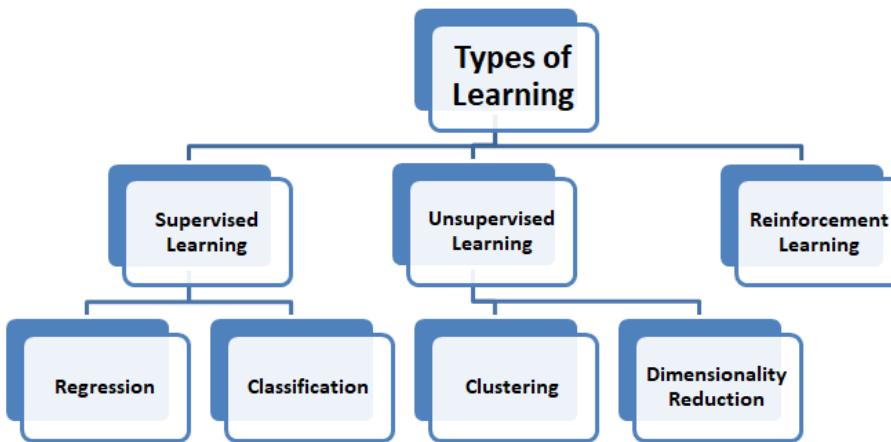
def rank_docs(query, docs, model):
    q_vec = model.encode([query])
    d_vecs = model.encode(docs)
    scores = cosine_similarity(q_vec, d_vecs)[0]
    ranked = sorted(zip(docs, scores), key=lambda x: x[1], reverse=True)
    return ranked[:3]

```

Machine Learning

Types of Machine Learning

- Supervised:** learns from labeled data (e.g., text classification).
- Unsupervised:** discovers hidden structure (e.g., clustering).
- Reinforcement:** learns from feedback/reward.
- NLP tasks map naturally to these paradigms.



(Ref: <https://www.studytrigger.com/article/types-of-learning-in-machine-learning/>)

Classification

- Predicts categorical labels (spam, positive/negative sentiment).
- Logistic Regression is a strong baseline for text classification.

```

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

vec = TfidfVectorizer()
X_train = vec.fit_transform(train_texts)
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
preds = model.predict(vec.transform(test_texts))
  
```

Clustering

- Groups similar documents based on content.
- Common methods: KMeans, DBSCAN, Agglomerative.
- Supports topic discovery and unsupervised organization.

```

from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

km = KMeans(n_clusters=5, random_state=42)
km.fit(embeddings)
labels = km.labels_

pca = PCA(n_components=2)
reduced = pca.fit_transform(embeddings)
plt.scatter(reduced[:,0], reduced[:,1], c=labels)
  
```

`plt.show()`

Project: Iris Flower Classifier

- Classic ML dataset demonstration.
- Illustrates feature scaling, model training, and evaluation.

```

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
model = LogisticRegression(max_iter=300)
model.fit(X_train, y_train)
print("Accuracy:", accuracy_score(y_test, model.predict(X_test)))
  
```

Project: Document Clusterer

- Embeds documents using spaCy or sentence-transformers.
- Applies KMeans for unsupervised grouping.
- Visualizes using t-SNE or PCA for semantic clusters.

```

import spacy
from sklearn.cluster import KMeans
nlp = spacy.load("en_core_web_md")

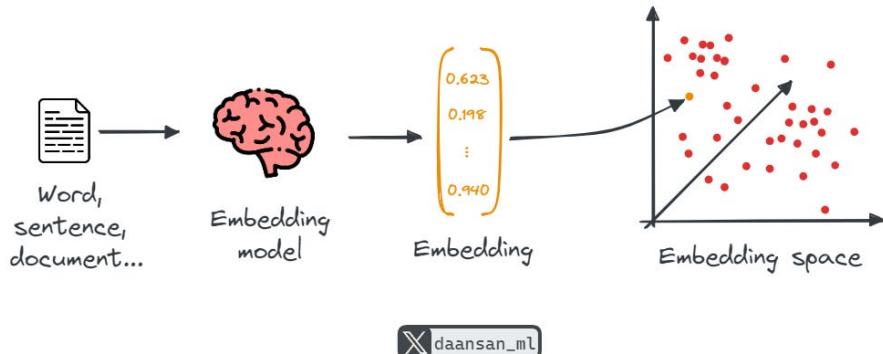
docs = ["AI in healthcare", "Quantum computing basics", "AI in finance"]
vectors = [nlp(d).vector for d in docs]
labels = KMeans(n_clusters=2).fit_predict(vectors)
print(list(zip(docs, labels)))
  
```

Word Embeddings

Word Embeddings Concepts

- Represent words as dense numerical vectors capturing context.
- Models: Word2Vec, GloVe, FastText, Transformer embeddings.
- Enable semantic understanding and transfer learning.

Embeddings



(Ref: <https://mlpills.substack.com/p/issue-58-embeddings-in-nlp>)

Project: Semantic Search Engine

- Retrieve documents semantically, not just by keyword match.
- Uses embedding models and cosine similarity for ranking.

```
from sentence_transformers import SentenceTransformer, util
model = SentenceTransformer("all-MiniLM-L6-v2")

def semantic_search(query, docs):
    q_emb = model.encode(query, convert_to_tensor=True)
    d_emb = model.encode(docs, convert_to_tensor=True)
    scores = util.cos_sim(q_emb, d_emb)
    return sorted(zip(docs, scores[0]), key=lambda x: float(x[1]),
    reverse=True)[:3]
```

Project: Word Analogy Solver

- Demonstrates vector arithmetic: $king - man + woman \approx queen$
- Captures latent gender, tense, and semantic relations.

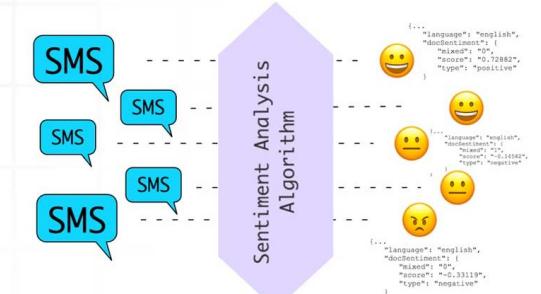
```
from gensim.models import KeyedVectors
model = KeyedVectors.load_word2vec_format("GoogleNews-vectors.bin", binary=True)
result = model.most_similar(positive=["king", "woman"], negative=["man"],
    topn=1)
print(result) # [('queen', 0.75)]
```

Sentiment Analysis

Sentiment Analysis Basics

- Detects emotional polarity (positive, neutral, negative).
- Used in reviews, feedback, and social media monitoring.
- Can be rule-based, ML-based, or transformer-based.

What is Sentiment Analysis?



(Ref: Sentiment Analysis — Engati)

Project: Review Sentiment Analyzer

- Train a sentiment classifier using Naive Bayes.
- Vectorize text with TF-IDF.
- Evaluate accuracy using test set.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

X = vec.fit_transform(reviews)
model = MultinomialNB()
model.fit(X, labels)
preds = model.predict(vec.transform(test_reviews))
print("Accuracy:", accuracy_score(test_labels, preds))
```

Project: Tweet Sentiment Dashboard

- Streams tweets and classifies their sentiment in real-time.
- Visualizes sentiment trends dynamically using dashboards.

```
import tweepy, pandas as pd
from textblob import TextBlob

def classify_sentiment(text):
    polarity = TextBlob(text).sentiment.polarity
    return "positive" if polarity > 0 else "negative" if polarity < 0 else
    "neutral"

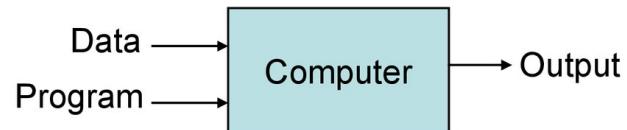
# TODO: integrate with dashboard for visualization
```

LLMs

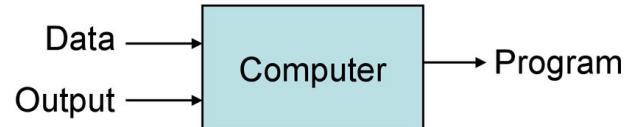
Background

Traditional vs. Machine Learning?

Traditional Programming



Machine Learning

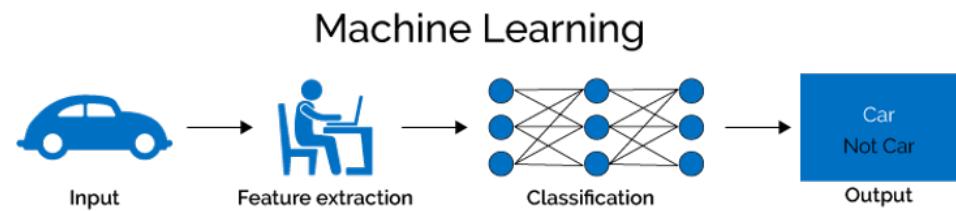


Why Machine Learning?

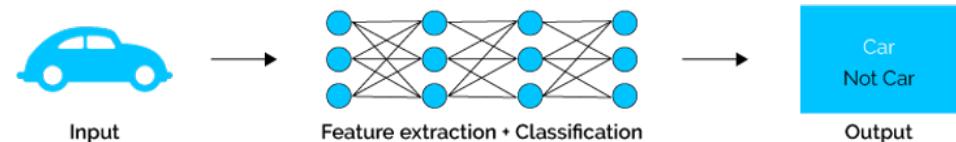
- Problems with High Dimensionality
- Hard/Expensive to program manually
- Job \$\$\$

ML vs DL: What's the difference?

Deep learning algorithms attempt to learn (multiple levels of) representation by using a hierarchy of multiple layers



Deep Learning



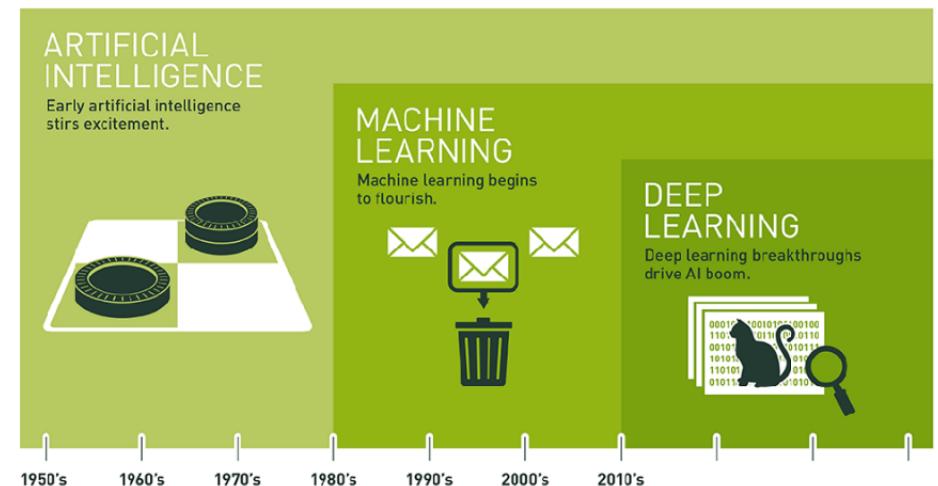
(Reference: <https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png>)

Use Deep Learning When ...

- You have lots of data (about 10k+ examples)
- The problem is “complex” - speech, vision, natural language
- The data is unstructured
- Techniques to model ‘ANY’ function given ‘ENOUGH’ data.

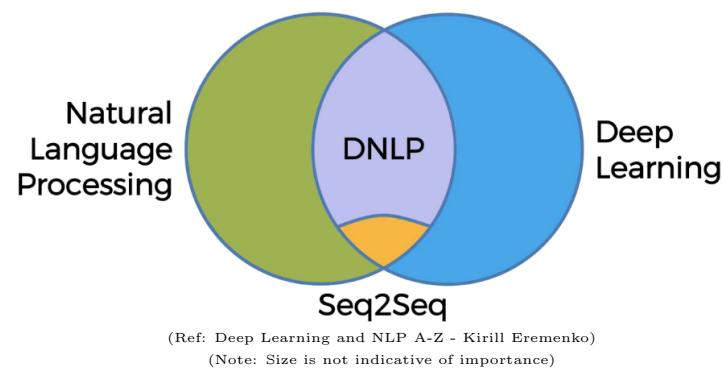
(Ref: Introduction to TensorFlow 2.0 - Brad Miro)

Relationship between AI, ML, DL



(Ref: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>)

What is Deep NLP



Seq2Seq is heavily used technique of DNLP for sequence to sequence modeling, eg Translation, Q & A, etc. Thats the basis of Large Language Models (LLMs)

Overview of Large Language Models

Typical Machine Learning Classification

- Each text item thus gets converted to fixed size vector, thus features.
- In training, weights are computed based on the given target.
- Once model is ready, it is able to answer target, say, Yes or No to unseen text.

Hello Kirill, Checking if you are back to Oz. Let me know if you are around ... Cheers, V



Training Data:

Hey mate, have you read about Hinton's capsule networks?	→	No
Did you like that recipe I sent you last week?	→	Yes
Hi Kirill, are you coming to dinner tonight?	→	Yes
Dear Kirill, would you like to service your car with us again?	→	No
Are you coming to Australia in December?	→	Yes

(Ref: Deep Learning and NLP A-Z - Kirill Eremenko)

Evolution of Vectorization

Vectors can be statistical (frequency based) or Machine/Deep Learning (supervised) based. Simple to complex.



(Ref: Analytics Vidhya https://editor.analyticsvidhya.com/uploads/59483evolution_of_NLP.png)

How to Vectorize? Representing words by their context



- **Distributional semantics:** A word's meaning is given by the words that frequently appear close-by
 - “*You shall know a word by the company it keeps*” (J. R. Firth 1957)
 - One of the most successful ideas of modern statistical NLP!
- When a word w appears in a text, its **context** is the set of words that appear nearby (within a fixed-size window).
- Use the many contexts of w to build up a representation of w

...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**

(Ref: CS224n: Natural Language Processing with Deep Learning - Christopher Manning)

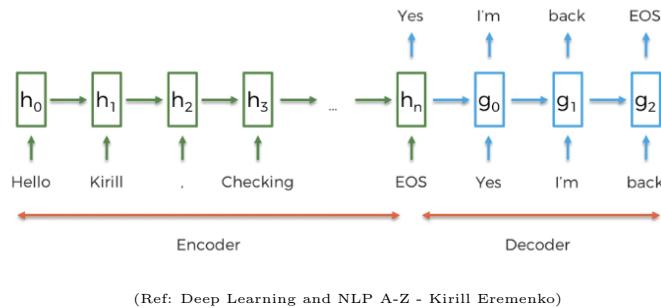
Word vectors

- Dense vector for each word
- Called distributed representation, word embeddings or word representations
- Test: similar to vectors of words that appear in similar contexts

$$\text{banking} = \begin{cases} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{cases}$$

Seq2Seq architecture

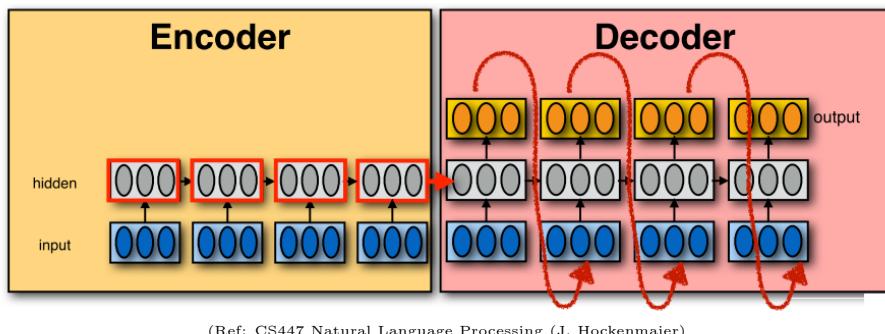
Hello Kirill. Checking if you are back to Oz.



During training, Encoder is fed with Questions and decoder with Answers. Weights in gates, hidden states get settled. During testing for each sequence of input, encoder results in to a combo vector. Decoder takes this and starts spitting out words one by one, probabilistically.

Encoder-Decoder (seq2seq) model

- The decoder is a language model that generates an output sequence conditioned on the input sequence.
 - Vanilla RNN: condition on the last hidden state
 - Attention: condition on all hidden states

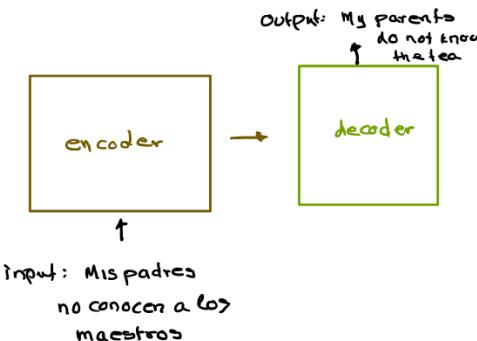


Transformers use Self-Attention

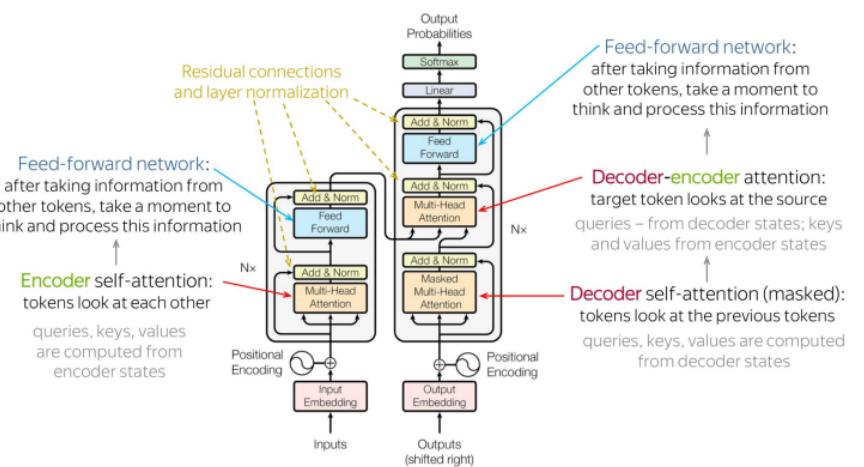
- Attention so far (in seq2seq architectures): In the decoder (which has access to the complete input sequence), compute attention weights over encoder positions that depend on each decoder position
- Self-attention: If the encoder has access to the complete input sequence, we can also compute attention weights over encoder positions that depend on each encoder position

Transformers

- In its heart it contains an encoding component, a decoding component, and connections between them.
- The Transformer is a model that uses attention to boost the speed with which seq2seq with attention models can be trained.
- The biggest benefit, however, comes from how The Transformer lends itself to parallelization. How?



Transformer Models

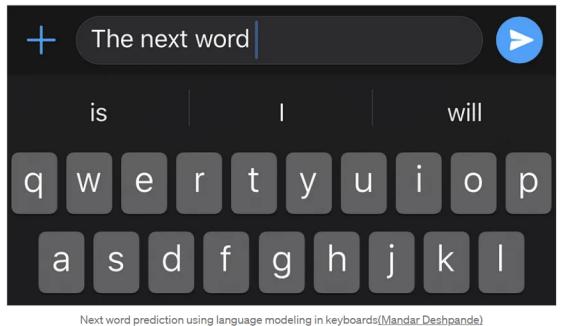


- No recurrence, so parallelization possible
- Context information captured via attention and positional encodings
- Consists of stacks of layers with various sublayers

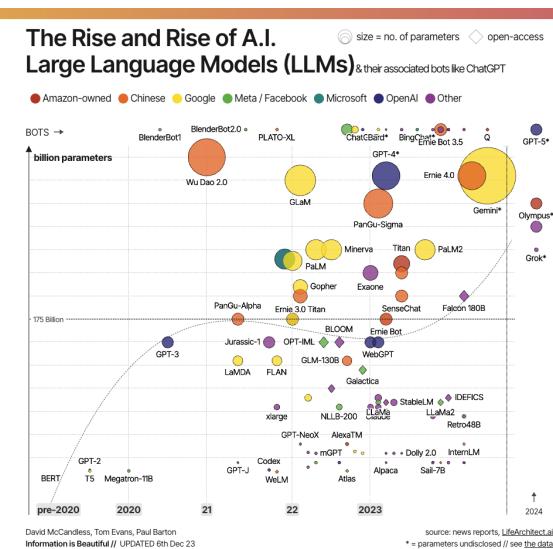
Transformers are basis of (the most) Large Language Models

What is a Language Models?

- While typing SMS, have you seen it suggests next word?
- While typing email, have you seen next few words are suggested?
- How does it suggest? (suggestions are not random, right?)
- In the past, for “Lets go for a …”, if you have typed ‘coffee’ 15 times, ‘movie’ say 4 times, then it learns that. Machine/Statistical Learning.
- Next time, when you type “Lets go for a ”, what will be suggested? why?
- This is called Language Model. Predicting the next word. When done continuously, one after other, it spits sentence, called Generative Model.



LLM - Information is beautiful



(Ref: <https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-langs-like-chatgpt/>)

How LLMs work?

- **Transformer-Based Architecture:**
 - Utilizes the Transformer architecture for processing input sequences.
 - Self-attention mechanism captures long-range dependencies.
- **Pre-training:**
 - Trained on a massive corpus of text data in an unsupervised manner.
 - Learns contextualized representations of words and phrases.
- **Generative Capabilities:**
 - Can generate coherent and contextually relevant text.
 - Useful for a wide range of natural language understanding and generation tasks.
- **Fine-tuning (Optional):**
 - Model can be fine-tuned on specific downstream tasks.
 - Adaptation to user or domain-specific requirements.

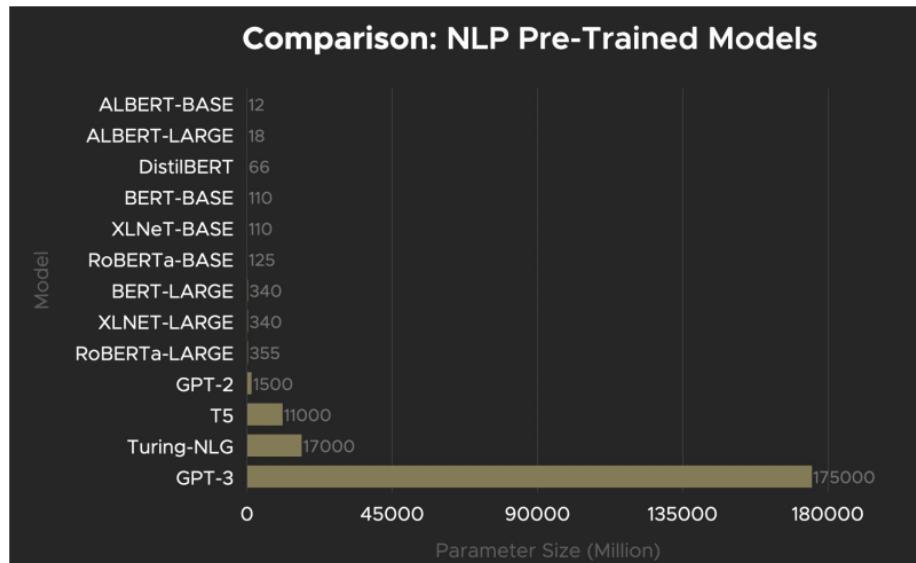
Transformer Architecture

- **Input Representation:** Embedding layer converts input tokens into high-dimensional vectors.
- **Positional Encoding:** Adds positional information to the input embeddings.
- **Multi-Head Self Attention:**
 - Allows each token to focus on different parts of the input sequence.
 - Multiple attention heads capture diverse patterns.
- **Layer Normalization & Residual/Skip Connections:**
 - Stabilizes training using layer normalization.
 - Residual connections help in mitigating vanishing/exploding gradient problems.
- **Encoder-Decoder Structure (for Sequence-to-Sequence tasks):** In tasks like translation, multiple encoder layers process the input, and then multiple decoder layers generate the output.
- **Output Layer:** Produces the final output sequence.

Decoder-Only Transformers (e.g., GPT)

- **Architecture:**
 - GPT uses a transformer architecture consisting solely of decoder layers.
 - Decoders attend to the entire input sequence during training and generation.
- **Positional Embeddings:** Incorporates position information to handle sequence order.
- **Self-Attention Mechanism with Masking:**
 - Each position attends to all positions in the preceding context.
 - During training, masking ensures that positions after the current one are not considered.
 - Prevents the model from peeking at future tokens during generation.
- **Autoregressive Generation:**
 - Generates output tokens one at a time in an autoregressive manner.
 - Previous tokens influence the generation of subsequent tokens.

Large Language Models - Comparison



(Ref: Deus.ai <https://www.deus.ai/post/gpt-3-what-is-all-the-excitement-about>)

LLM Training

- Training LLMs involves instructing the model to comprehend and generate human-like text.
- Input Text:** LLMs are exposed to extensive text data from diverse sources like books, articles, and websites.
- During training, the model predicts the next word/token based on context, learning patterns and relationships.
- Optimizing Weights:** The model has weights for parameters reflecting feature significance.
- Throughout training, weights are fine-tuned to minimize error, enhancing the model's prediction accuracy.
- After initial training, LLMs can be customized for tasks using small sets of supervised data. This process is known as **fine-tuning**.
- Fine-tuning Parameters:** LLMs adjust parameter values based on error feedback during predictions.
- The model refines its language understanding by iteratively adjusting parameters, improving token prediction accuracy.
- Training may vary for specific LLM types, like those optimized for continuous text or dialogue.

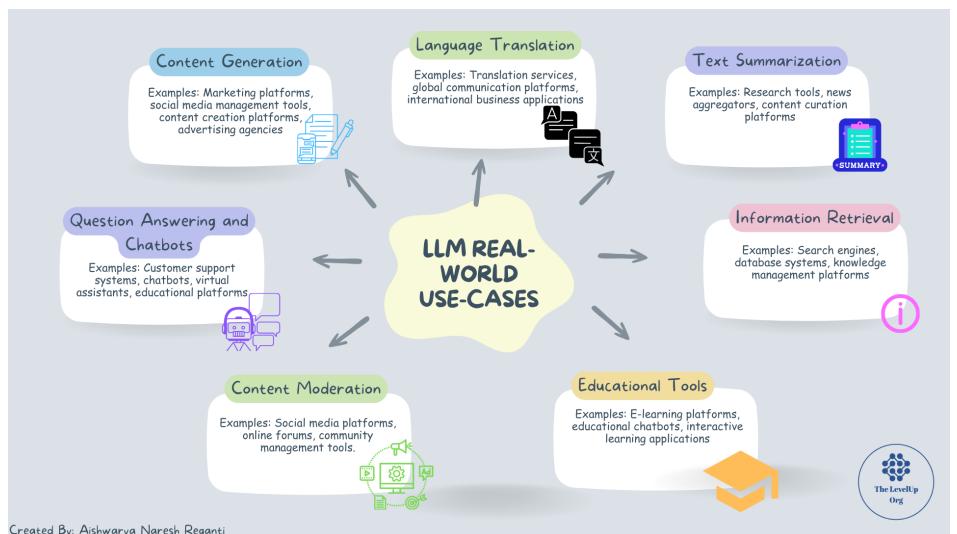
LLM Performance Factors

- Model Architecture:** LLM performance is influenced by the design and intricacy of its architecture.
- Dataset Quality:** The quality and diversity of the training dataset shape the model's language understanding.
- Training a private LLM demands substantial computational resources and expertise.
- Duration ranges from days to weeks, contingent on model complexity and dataset size.
- Cloud-based solutions and high-performance GPUs expedite the training process.
- LLM training is meticulous and resource-intensive, forming the basis for language comprehension and generation.

Decoder-Only Transformers (e.g., GPT)

- Architecture:**
 - GPT uses a transformer architecture consisting solely of decoder layers.
 - Decoders attend to the entire input sequence during training and generation.
- Positional Embeddings:** Incorporates position information to handle sequence order.
- Self-Attention Mechanism with Masking:**
 - Each position attends to all positions in the preceding context.
 - During training, masking ensures that positions after the current one are not considered.
 - Prevents the model from peeking at future tokens during generation.
- Autoregressive Generation:**
 - Generates output tokens one at a time in an autoregressive manner.
 - Previous tokens influence the generation of subsequent tokens.

LLM Real World Use Cases



(Ref: Applied LLMs Mastery 2024 - Aishwarya Reganti)

LLM Real World Use Cases

No.	Use case	Description
1	Content Generation	Craft human-like text, videos, code and images when provided with instructions
2	Language Translation	Translate languages from one to another
3	Text Summarization	Summarize lengthy texts, simplifying comprehension by highlighting key points.
4	Question Answering and Chatbots	LLMs can provide relevant answers to queries, leveraging their vast knowledge
5	Content Moderation	Assist in content moderation by identifying and filtering inappropriate or harmful language
6	Information Retrieval	Retrieve relevant information from large datasets or documents.
7	Educational Tools	Tutor, provide explanations, and generate learning materials.

(Ref: Applied LLMs Mastery 2024 - Aishwarya Reganti)

LLM Challenges



(Ref: Applied LLMs Mastery 2024 - Aishwarya Reganti)

ChatGPT — GPT3.5/GPT4

Playground

PPC - Long Descriptions

Model: text-davinci-003

Temperature: 0.7

Maximum length: 1750

Stop sequences: Enter sequence and press Tab

Top P: 1

Frequency penalty: 0.2

Presence penalty: 0.45

Best of: 12

Inject start text:

Inject restart text:

Looking for ChatGPT? Try it now!

(Ref: The Complete Prompt Engineering for AI Bootcamp (2023))

Created by OpenAI Access it with code or without (Playground <https://platform.openai.com/playground>)

Bard — Palm 2/Gemini

Product description: A home milkshake maker
Seed words: fast, healthy, compact
Product names: HomeShaker, Fit Shaker, QuickShake, Shake Maker

Product description: A pair of shoes that can fit any foot size.
Seed words: adaptable, fit, omni-fit
Product names:

Here are some product names for a pair of shoes that can fit any foot size:

- Omni-Fit Shoes
- Adaptable Shoes
- Fit-All Shoes
- Size-Free Shoes
- Universal Shoes

(Ref: The Complete Prompt Engineering for AI Bootcamp (2023))

Created by Google Access it via chat <https://bard.google.com/> or encounter it in search results

Meta LLaMA

- Open-Source. Need to build a UX and any advanced functionality around it, and may need to fine-tune it.
- Many use-cases in the enterprise can't use OpenAI for fear of sensitive data leaking or being used to train the model (though OpenAI claims to keep API data private).
- If you have 200+ examples fine-tuning beats prompt engineering for a specific defined task.

(Ref: The Complete Prompt Engineering for AI Bootcamp (2023))

Anthropic Claude

The screenshot shows a user interface for generating product names. At the top, there's a header with a square icon, three dots, and tabs for 'Chat' and 'Edit Chat'. Below this, a section titled 'Fit Shaker: Fast and Healthy' contains seed words: 'adaptable, fit, omni-fit'. A text input field says 'Product names:' followed by a blue circular button with a white 'M'. A purple sidebar lists suggested product names: 'OmniFit', 'AdaptiShoe', 'UniSole', 'One Size Fits All', and 'FlexiFit'. At the bottom, a message input field says 'Write a message...' with a blue send button containing a white arrow.

(Ref: The Complete Prompt Engineering for AI Bootcamp (2023))

Created by Anthropic <https://console.anthropic.com/> or API Uses Constitutional AI rather than RLHF

Constitutional AI trains to follow a set of high-level principles or rules, such as a constitution, that specify the desired behavior and outcomes of the system. RLHF uses human feedback, such as ratings, preferences, or corrections, to optimize a language model or an agent's policy using reinforcement learning

Microsoft Bing — GPT 4

The screenshot shows a user interface for planning a trip to London. At the top, a blue bar says 'Help me plan a trip to London'. Below it, a green checkmark says 'Generating answers for you...'. The main text area says: 'Sure, I can help you plan a trip to London. London is a great city to visit, with iconic attractions, historic museums, and vibrant neighborhoods. You can find things to do in London, from sightseeing tours to fun-filled days out, on [visitlondon.com](#) ¹. You can also book a London Pass to make the most of popular attractions ² or a one-day hop-on hop-off bus tour to explore the sights at your own pace ³. You can find cheap flights to London from \$99 on KAYAK ⁴ and London vacation packages from \$849 on Expedia ⁵.'. At the bottom, a blue bar says 'Learn more: [1. visitlondon.com](#) +4 more' and 'Ask me anything' with a microphone icon.

(Ref: The Complete Prompt Engineering for AI Bootcamp (2023))

Powered by OpenAI's GPT-4 <https://www.microsoft.com/en-gb/bing>

Falcon

Downloads last month
158,979



⚡ Hosted inference API ⓘ

Text Generation

Example 8 ▾

Once upon a time, there was a little girl named Alice. She was a curious little girl, who was always asking questions|

Compute

⌘+Enter

1.5

Computation time on gpu: cached

⤵ JSON Output

Maximize

(Ref: The Complete Prompt Engineering for AI Bootcamp (2023))

Access it via HuggingFace transformers library, 7B and 40B models as well as instruct fine-tuned Features:

- Free for commercial use
- Open source
- Possible to fine-tune

Leader board (Jan 2024)

Leaders by category

The leaderboard ranks LLMs across multiple prompt categories.

RANK ↗	LLM	TOTAL	BRAINSTORMING	CLOSED QA	GENERATION	OPEN QA	REWRITE
1	GPT-4	81.75	89.74	78.95	81.87	74.10	92.31
2	WizardLM 13B V1.2	79.56	76.92	73.68	80.31	77.11	92.31
3	LLaMA 2 70B Chat	78.97	88.46	68.42	81.35	69.88	84.62
4	GPT-3.5 Turbo	76.79	73.08	68.42	80.31	73.49	76.92
5	Vicuna 33B V1.3	74.21	82.05	47.37	71.50	70.48	76.92
6	Guanaco 13B	50.00	50.00	50.00	50.00	50.00	50.00

(Ref: <https://toloka.ai/llm-leaderboard/>)

Want to give it a try? - Hugging Face APIs

(Ref: What are Large Language Models(LLMs)? -Suvrojit Hore)

Sentence Completion

```
import requests
from pprint import pprint

API_URL = 'https://api-inference.huggingface.co/models/bigscience/bloomz'
headers = {'Authorization': 'Entertheaccesskeyhere'}

def query(payload):
    response = requests.post(API_URL, headers=headers, json=payload)
    return response.json()

params = {'max_length': 200, 'top_k': 10, 'temperature': 2.5}
output = query({
    'inputs': 'Sherlock Holmes is a',
    'parameters': params,
})

print(output)

[{'generated_text': 'Sherlock Holmes is a private investigator whose cases ',
 'have inspired several film productions'}]
```

Question Answers

```
API_URL =
    'https://api-inference.huggingface.co/models/deepset/roberta-base-squad2'
headers = {'Authorization': 'Entertheaccesskeyhere'}

def query(payload):
    response = requests.post(API_URL, headers=headers, json=payload)
    return response.json()

params = {'max_length': 200, 'top_k': 10, 'temperature': 2.5}
output = query({
    'inputs': {
        'question': "What's my profession?",
        'context': "My name is Yogesh and I am an AI Coach"
    },
    'parameters': params
})

pprint(output)

{'answer': 'AI Coach',
 'end': 39,
 'score': 0.7751647233963013,
```

```
'start': 30}
```

Summarization

```
API_URL = "https://api-inference.huggingface.co/models/facebook/bart-large-cnn"
headers = {'Authorization': 'Enter the access key here'}

def query(payload):
    response = requests.post(API_URL, headers=headers, json=payload)
    return response.json()

params = {'do_sample': False}

full_text = '''AI applications are summarizing articles, writing stories and
engaging in long conversations and large language models are doing
the heavy lifting.

'''

output = query({
    'inputs': full_text,
    'parameters': params
})
print(output)

[{'summary_text': 'Large language models - most successful
    'applications of transformer models. ...'}]
```

Conclusions, Cautions and What's Next?

So, What are LLMs?

Large Language Models (LLMs) have revolutionized natural language processing, ushering in advancements in text generation and understanding. Key attributes include:

- **Learning from Extensive Data:** LLMs acquire knowledge from vast datasets, resembling a massive library of information.
- **Grasping Context and Entities:** These models understand context and entities, allowing for a deeper comprehension of language.
- **Proficient User Query Responses:** LLMs excel in responding to user queries, showcasing their ability to apply learned knowledge effectively.

Despite their versatile applications across industries, ethical concerns and potential biases necessitate a critical evaluation to understand their societal impact.

Core Beliefs of Large Language Models

- No inherent "core beliefs."
- Word guessers predicting internet-like sentences.
- Can write both for and against a topic without belief.
- Emulates the most common response in training data.

Truth and Morality in Large Language Models

- Lack sense of truth or morality.
- Tendency to generate words we agree are true.
- No guarantee of providing the actual truth.

Mistakes in Large Language Models

- Prone to mistakes due to inconsistent training data.
- Self-attention may not capture all relevant information.
- Hallucination: generating words not derived from input.
- Preference for common words, small numbers, and specific names.

Auto-regressive Nature of LLMs

- Auto-regressive models: guesses affect subsequent inputs.
- Errors accumulate, potentially compounding mistakes.
- No mechanism to "change minds" or self-correct.
- Lack the ability to retry or undo prior choices.

Verification of Outputs

- Always verify outputs of large language models.
- Assess competence to verify results in high-stakes tasks.
- Mistakes in critical tasks may lead to costly decisions.

Input Size and Memory Limitations

- Large language models have input size limits.
- Conversation appears coherent until log size exceeds limit.
- Earlier parts of the conversation are deleted, and the model "forgets."

Ethical Considerations

- Awareness of potential biases in LLMs is crucial for responsible usage.
- Continuous evaluation of ethical implications is necessary to mitigate societal risks.
- Balancing the benefits of LLMs with ethical concerns ensures responsible deployment.

Future Impact

- LLMs expected to revolutionize domains such as job markets, communication, and society.
- Careful use and ongoing development are essential for positive impacts.
- Understanding limitations and ethical considerations is vital for responsible integration into various domains.

Landscape of LLMs & Quiz

- Types of models - Foundation models, LLM, SLM, VLMs, etc.
- Common LLM terms - Prompts, Temperature, Hallucinations, Tokens, etc.
- LLM lifecycle stages - Pre-training, Supervised Fine Tuning, RLHF, etc.
- LLM evaluations - ROUGE, BLEU, BIG-bench, GLUE, etc.
- LLM architecture - Encoder, Decoder, Transformer, Attention, etc.
- Retrieval augmented generation - Vector DBs, Chunking, Evaluations, etc.
- LLM agents - Memory, Planning, ReAct, CoT, ToT, etc.
- Cost efficiency - GPU, PEFT, LoRA, Quantization, etc.
- LLM security - Prompt Injection, Data poisoning, etc.
- Deployment & inference - Pruning, Distillation, Flash Attention, etc.
- Platforms supporting LLMOps

(Ref: LinkedIn post by Abhinav Kimothi - 23 Jan 2024)

Conclusions

Conclusions

- Robust NLP preprocessing and embeddings are critical for effective information retrieval.
- spaCy and scikit-learn streamline NLP + ML workflows.
- Vector-based understanding enables semantic and contextual search.
- Combining these components forms the backbone of Retrieval-Augmented Generation.
- Future directions:
 - Fine-tuning LLMs for domain tasks
 - Efficient vector database retrieval (FAISS, Milvus)
 - Integrating sentiment and entity-level reasoning

References

- Explosion AI, *spaCy Documentation*, <https://spacy.io/>
- Pedregosa et al., *Scikit-learn: Machine Learning in Python*, JMLR, 2011.
- Mikolov et al., *Efficient Estimation of Word Representations in Vector Space*, arXiv, 2013.
- Bird et al., *Natural Language Processing with Python*, O'Reilly, 2009.
- OpenAI, *Retrieval-Augmented Generation (RAG) Overview*, 2024.
- Manning & Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press.
- Jurafsky & Martin, *Speech and Language Processing*, 3rd Ed. Draft, 2023.