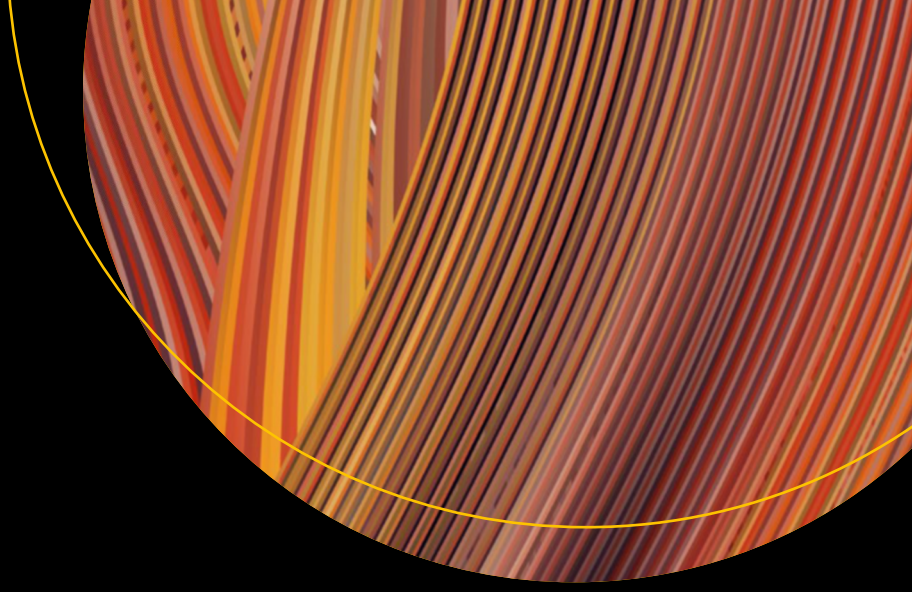

Introduction to Python and Large Language Models

A Guide to Language Models

Dilyan Grigorov

Apress®

Introduction to Python and Large Language Models

A Guide to Language Models

Dilyan Grigorov

Apress®

Introduction to Python and Large Language Models: A Guide to Language Models

Dilyan Grigorov
Varna, Bulgaria

ISBN-13 (pbk): 979-8-8688-0539-4 ISBN-13 (electronic): 979-8-8688-0540-0
https://doi.org/10.1007/979-8-8688-0540-0

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Celestin Suresh John
Development Editor: James Markham
Coordinating Editor: Gryffin Winkler

Cover designed by eStudioCalamar

Cover image by flatart on Freepik.com

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub (https://github.com/Apress). For more detailed information, please visit https://www.apress.com/gp/services/source-code.

If disposing of this product, please recycle the paper

This book is dedicated to all my mentors at Google,
who not only sparked my passion for coding but also inspired
me to continuously pursue it without halt.

Table of Contents

About the Author

Dilyan Grigorov is a software developer with a passion for Python software development, generative deep learning and machine learning, data structures, and algorithms. He is an advocate for open source and the Python language itself. He has 16 years of industry experience programming in Python and has spent five of those years researching and testing generative AI solutions. Dilyan is a Stanford student in the Graduate Program on Artificial Intelligence in the classes of people like Andrew Ng, Fei-Fei Li, and Christopher Manning. He has been mentored by software engineers and AI experts from Google and Nvidia. His passion for AI and ML stems from his background as an SEO specialist dealing with search engine algorithms daily. He enjoys engaging with the software community, often giving talks at local meetups and larger conferences. In his spare time, he enjoys reading books, hiking in the mountains, taking long walks, playing with his son, and playing the piano.

About the Technical Reviewer

Tuhin Sharma is Sr. Principal Data Scientist at Red Hat in the Data Development Insights and Strategy group. Prior to that, he worked at Hypersonix as an AI architect. He also co-founded and has been CEO of Binaize, a website conversion intelligence product for e-commerce SMBs. He received a master's degree from IIT Roorkee and a bachelor's degree from IIEST Shibpur in Computer Science. He loves to code and collaborate on open source and research projects. He has four research papers and five patents in the field of AI and NLP. He is a reviewer of IEEE MASS conference in the AI track. He writes deep learning articles for O'Reilly in collaboration with the AWS MXNET team. He is a regular speaker at prominent AI conferences like O'Reilly Strata & AI, ODSC, GIDS, Devconf, etc.

Acknowledgments

As I reflect on the journey of writing this book, I am overwhelmed with gratitude for the countless individuals who have supported, inspired, and guided me along this path.

First and foremost, I extend my deepest thanks to my family, whose unwavering support and endless patience have been my anchor and source of strength.

I owe a great debt of gratitude to my mentors and colleagues, who have shared their wisdom, critiqued my ideas with kindness, and encouraged me to push the boundaries of my knowledge. Special thanks to my mentor Alexandre Blanchette whose insightful feedback and encouragement were invaluable to the completion of this manuscript.

Very special thanks to another of my mentors – Haiguang Li, one of the best machine learning and AI experts I have ever met. You have been my north star throughout this writing process. Your belief in me has been a gift beyond measure. Your generous sharing of knowledge, patience, and encouragement has not just shaped this book but has transformed me as a writer, software engineer, and individual. My sincerest thanks for your invaluable contribution.

My appreciation extends to the team at Apress and Springer Group, especially my editor, Celestine Suresh John, whose keen eye and creative vision have significantly enhanced this book. Thank you for your patience, guidance, and commitment to excellence.

A special word of thanks goes to the teams behind the development of large language models. To the researchers and engineers at organizations like OpenAI, Google Brain, the Google Research team, and others who have pushed the boundaries of what's possible with AI and machine learning, and who generously share their findings and tools with the world. Your work has not only inspired this book but also revolutionized how we think about human–computer interaction.

Finally, to you, the reader, embarking on this journey to explore Python and large language models: I hope this book serves as a valuable guide and inspires you to delve deeper into the transformative power of programming and AI. Thank you for your curiosity and your commitment to learning!

Introduction

In the evolving landscape of technology, where the boundaries between science fiction and reality blur, lies a transformative tool: the large language model (LLM). These models, sophisticated engines of artificial intelligence, have not only redefined our interaction with machines but have also opened new avenues for understanding human language. This book, structured into seven comprehensive chapters, serves as both a beacon and a bridge for those embarking on a journey through the intricate world of LLMs and their application using Python.

Chapter 1, "Evolution and Significance of Large Language Models," lays the foundation. It's here we start our journey, unraveling the complex yet fascinating world of natural language processing (NLP) and large language models. With over 50 pages dedicated to setting the stage, this chapter aims to provide the reader with a solid understanding of the evolution, significance, and basic concepts underpinning LLMs. Through a meticulous exploration of topics such as text preprocessing, word embeddings, and sentiment analysis, we uncover the magic and mechanics of LLMs and their impact across various domains.

Chapter 2, "What Are Large Language Models?", shifts the focus to the tools that make working with LLMs possible, with a particular emphasis on "Python and Why Python for LLMs?" It demystifies Python – a language synonymous with simplicity and power in the world of programming. From basic syntax to the nuanced features of Python 3.11, readers will gain the necessary knowledge to navigate the subsequent chapters and harness Python for their LLM endeavors.

In Chapter 3, "Python for LLMs," we plunge into the heart of LLMs, dissecting their components and understanding their workings. This chapter covers everything from embedding layers to attention mechanisms, providing insights into the technical makeup of models like GPT-4, BERT, and others. It's a chapter designed to equip readers with a profound understanding of how LLMs predict the next token, learn from few examples, and, occasionally, hallucinate.

Chapter 4, "Python and Other Programming Approaches," is a practical guide to leveraging Python for LLM development. Here, readers will familiarize themselves with essential Python libraries, frameworks, and platforms such as Hugging Face and OpenAI

API, exploring their use in building applications powered by LLMs. With a focus on data preparation and a showcase of basic examples built with each framework, this chapter is a testament to Python's role in the democratization of AI.

Chapter 5, "Basic Overview of the Components of the LLM Architectures," demonstrates the versatility and potential of LLMs through practical Python applications. Readers will learn how to employ LLMs for tasks ranging from text generation to chatbots, each accompanied by step-by-step examples. This chapter not only highlights the capabilities of LLMs but also inspires readers to envision and create their own applications.

Chapter 6 of your document, titled "Applications of LLMs in Python," explores how Large Language Models (LLMs) are used in various domains, focusing on text generation and creative writing. It details how LLMs can generate human-like text using models like RNNs and transformers. The chapter covers key use cases, including content creation, chatbots, virtual assistants, and data augmentation. It also highlights how LLMs assist in creative writing tasks, brainstorming, dialogue crafting, world-building, and experimental literature. Additionally, the chapter discusses language translation, text summarization, and document understanding, emphasizing LLMs' impact on improving accuracy and efficiency in these areas. Finally, the chapter presents an example of building a question-answering chatbot using LLMs.

Chapter 7 explores how Python 3.11 and libraries such as LangChain, Hugging Face, and others are utilized to develop applications powered by large language models (LLMs). It covers the features of LangChain, such as model interaction, data connection, and memory, and explains how to build applications using these tools. The chapter also discusses the integrations and use cases of LangChain in various industries, like customer support, coding assistants, healthcare, and e-commerce, highlighting its flexibility in creating AI-powered solutions.

This book is an invitation to a world where understanding meets creation. It's for the curious minds eager to decode the language of AI and for the creators ready to shape the future. Whether you're a student, software engineer, data scientist, an AI or ML researcher or practitioner, or simply an enthusiast, the journey through these pages will equip you with the knowledge and skills to participate in the ongoing conversation between humans and machines. Welcome to the frontier of language, learning, and imagination.

CHAPTER 1

Evolution and Significance of Large Language Models

Over the recent decades, there have been notable advancements in language models and artificial intelligence technologies. Alongside advancements in computer vision, voice and speech processing, and image processing models, large language models (LLMs) are poised to profoundly impact the evolution of AI technologies. Therefore, it is crucial to examine the progress of language models since their inception and, more importantly, anticipate their future growth.

This chapter presents a concise overview of language models, tracing their development from statistical and rule-based models to today's transformer-based multimodal large language models with billions of parameters. It also aims to provide a good definition of what an LLM is and what are the mechanisms of work of the LLMs in general. Additionally, the benefits and limitations of existing models are observed and areas where current models require improvement are identified.

In recent times, substantial attention has been garnered by large language models, owing to numerous accomplishments in the field of natural language processing. Notably, the emergence of powerful chatbots like OpenAI's ChatGPT, Google Bard, and Meta's LLaMA, among others, has played a pivotal role. The achievements in language models are the culmination of decades of research and development. These models not only advance state-of-the-art NLP technologies but are also anticipated to significantly impact the evolution of AI technologies.

The foundational models, initially arising in NLP research, such as the transformers, have transcended into other domains like computer vision and speech recognition. However, AI models, especially language models, are not flawless, and the technology

D. Grigorov, *Introduction to Python and Large Language Models*,
https://doi.org/10.1007/979-8-8688-0540-0_1

itself is akin to a double-edged sword. There are unresolved aspects that require further research and analysis. It remains uncertain whether these models are sophisticated computer programs generating responses solely through numerical computations and probabilities or if they possess a form of understanding and intelligence.

The sheer size of these gigantic language models makes it challenging to interpret their internal logic meaningfully. Hence, understanding the history of language models is crucial for depicting a better picture of their future development. The following paragraphs aim to provide a concise overview of language models and their evolution. Their goal is to review key milestones and innovations in language and machine learning models that have significantly influenced today's modern large language models.

That's a matter of subject that's in front of us to research in the future:

- Firstly, it offers insights into the core engineering elements of powerful language models, aiding in understanding the nature of different variations.
- Then it presents a taxonomy of related research, highlighting various categories of language models, their shortcomings, and how these issues have been addressed over time.
- The hope is that this overview will offer valuable insights into the past, present, and future of language models, contributing to the development of trustworthy models aligned with universal human values.

The Evolutionary Steps of Large Language Models

This section acknowledges the foundational work of Andrey Markov and Claude Shannon in the early 20th century, introducing concepts like n-grams and information theory that laid the groundwork for language modeling. It also reviews Noam Chomsky's introduction of the Chomsky hierarchy of grammars in 1956, which addressed the limitations of finite-state grammars and advocated for the use of context-free grammars in NLP.

Markov, Shannon, and the Language Models

The inception of language models traces back to the contributions of **Andrey Markov**[1] **and Claude Shannon**[2] in the early and mid-20th century, respectively. Markov laid the mathematical groundwork for what we now recognize as n-grams. Intriguingly, the concept of n-grams was first proposed in Markov's earlier works. Andrey Markov stands out as one of the earliest scientists to delve into the study of language models, even though the term "language model" had yet to be coined during that period.

Consider the conditional probability denoted as `p(w1|w0) = p(w1)`. Various language models employ distinct approaches to compute conditional probabilities, denoted as `p(wi|w1, w2, ..., wi-1)`. The procedure of acquiring and applying a language model is known as **language modeling**. An n-gram model serves as a fundamental type of model, assuming that the word at any given position is solely influenced by the words at the preceding `n - 1` positions. In essence, the model operates as an **n – 1-order Markov chain**.

In 1906, Markov delved into the study of Markov chains, initially exploring a straightforward model with only two states and corresponding transition probabilities between them. He demonstrated that by traversing between these states based on the transition probabilities, the frequencies of transitioning to each state would ultimately converge to the expected values. Over subsequent years, Markov expanded the model and established that this conclusion held true in more generalized contexts.

Illustrating his model's practicality, Markov applied it to Alexander Pushkin's verse novel, "Eugene Onegin" in 1913. Through the removal of spaces and punctuation marks and the categorization of the first 20,000 Russian letters into vowels and consonants, Markov derived a sequence representing the novel's vowel and consonant distribution. Utilizing manual counting methods, Markov calculated the transition probabilities between vowels and consonants, using the resulting data to verify the characteristics of the simplest Markov chain.

[1] A. A. Markov, An Example of Statistical Investigation of the Text Eugene Onegin Concerning the Connection of Samples in Chains. www.alpha60.de/research/markov/,1913. 7., A. A. Markov, An example of statistical investigation in the text of `eugene onegin illustrating coupling `tests' in chains, Proceedings of the Academy of Sciences Of St.Petersburg, vol. 7,pp. 153–162, 1913

[2] C.E. Shannon, A mathematical theory of communication, The Bell System Technical Journal, vol. 27, no. 3, pp. 379–423, 1948

It is intriguing that the initial application area of the Markov chain was in language, with Markov's study serving as a foundational exploration into the **simplest form of a language model**.

Moving forward in the mid-20th century, **Claude Shannon** suggested the utilization of Markov processes to model natural language. Employing n-th order Markov chains, he developed a statistical model capable of characterizing the probability of sequences of letters, encompassing words and sentences. From a mathematical perspective, Shannon's approach involved counting the frequency of character sequences of length n, referred to as n-grams.

In 1948, Claude Shannon revolutionized the field of information theory with his seminal paper, "The Mathematical Theory of Communication." This work marked the inception of key concepts such as entropy and cross-entropy, as well as an exploration of the n-gram model. (Shannon adopted the term "entropy" from statistical mechanics following advice from John von Neumann.)

Entropy, in this context, signifies the uncertainty inherent in a probability distribution, while cross-entropy encapsulates the uncertainty of one probability distribution concerning another. Notably, entropy serves as a lower bound for cross-entropy. To elaborate, as the length of a word sequence tends toward infinity, the language's entropy can be defined, assuming a constant value that can be estimated from the language's data.

Shannon's contribution establishes a valuable tool for evaluating language models. If one language model demonstrates superior accuracy in predicting word sequences compared to another, it will exhibit lower cross-entropy. This insight provides a robust framework for assessing language modeling. It is essential to recognize that language models extend beyond natural languages to encompass formal and semi-formal languages.

The pioneering works of Markov and Shannon paved the way for diverse approaches to language modeling, encompassing rule-based systems, neural network–based systems, and more recently, **transformer-based pre-trained large language models** (LLMs) founded on attention mechanisms.

Chomsky and the Language Models

In 1956, Noam Chomsky[3] introduced the **Chomsky hierarchy of grammars**, a framework aimed at representing the **syntax of languages**. Chomsky emphasized the limitations of finite-state grammars, including n-gram models, in adequately describing natural languages.

Chomsky's theory posits that a language comprises a finite or infinite collection of sentences, where each sentence is a finite sequence of words drawn from a finite vocabulary. A grammar, according to Chomsky, is defined by a set of production rules capable of generating all sentences within the language. Different grammars yield languages of varying complexities, forming a **hierarchical structure**.

A grammar that can generate sentences acceptable by a finite-state machine is termed a finite-state grammar or regular grammar, whereas a grammar capable of producing sentences acceptable by a nondeterministic pushdown automaton is labeled a context-free grammar. Finite-state grammars are encompassed by context-free grammars.

The grammar underlying a finite Markov chain or an n-gram model aligns with a finite-state grammar. However, finite-state grammars face limitations in generating English sentences with grammatical relationships, such as those depicted in (i) and (ii).

- If S1, then S2.
- Either S3 or S4.
- Either if S5, then S6, or if S7, then S8.

While these relations can theoretically be combined indefinitely to create correct English expressions (as in example iii), finite-state grammars cannot account for all such combinations. Chomsky argued that there are significant constraints in describing languages, including n-gram models, with finite-state grammars. Instead, he advocated for the more effective modeling of languages using context-free grammars. This influence led to the increased adoption of context-free grammars in natural language processing (NLP) over the subsequent decades. Although Chomsky's theory may not wield significant influence in contemporary NLP, it retains important scientific value.

[3] N. Chomsky, Three models for the description of language. *IEEE Transactions on Information Theory 2*, 3 (1956), 113–124

Rule-Based Language Models

This section provides an overview of the first chatbot, ELIZA, created in 1966 by Joseph Weizenbaum. ELIZA's rule-based approach to dialogue simulation marked a significant milestone, although it lacked genuine comprehension of language nuances. This era saw a focus on rule-based chatbot programs, which led to the creation of several similar systems, including PARRY, ALICE, and CLEVER.

The First Chatbot: ELIZA

The exploration of language models traces back to the 1950s when researchers delved into **rule-based systems for language processing**. Initial attempts were hindered by the requirement to manually construct grammatical rules.

A significant breakthrough occurred in 1966 when Joseph Weizenbaum[4] introduced the "ELIZA" program (Figure 1-1). ELIZA stands out as one of the earliest computer programs capable of interacting with humans through chatting and question answering. Functioning as a simulated Rogerian psychotherapist, ELIZA engaged users in text-based conversations, representing a pioneering example of dialogue simulation. However, ELIZA's interactions relied heavily on predefined patterns and lacked genuine comprehension of the nuances in language.

[4] J. Weizenbaum, Eliza, a computer program for the study of natural language communication between man and machine, Communications of the ACM, vol. 9, no. 1, pp. 36–45, 1966

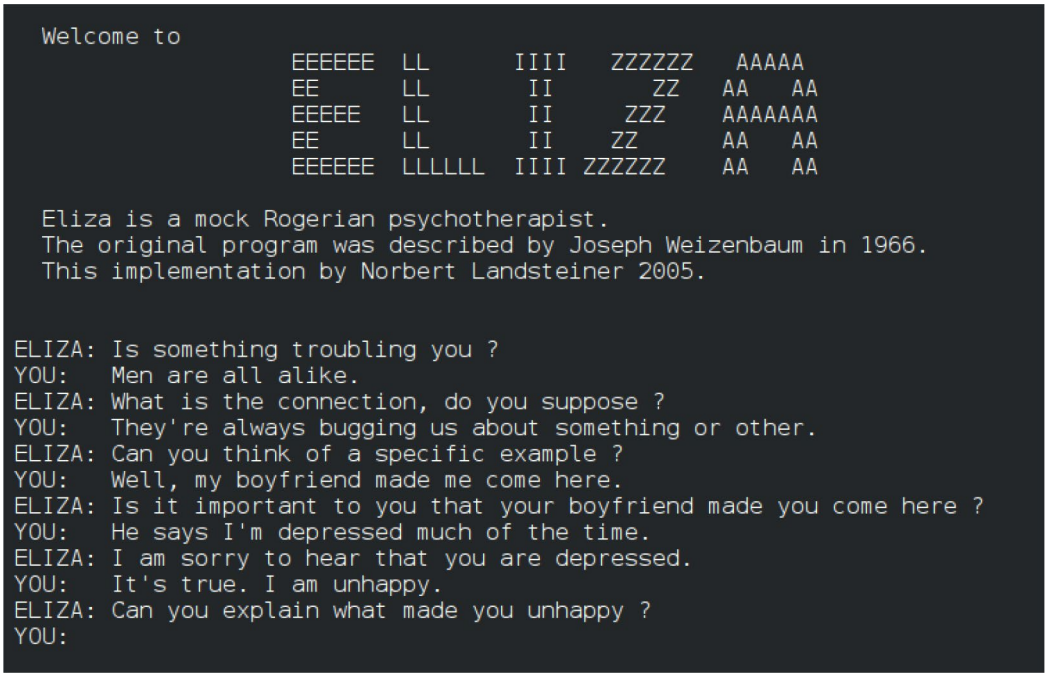

Figure 1-1. *"ELIZA" program, Source: Wikipedia*

Originating in 1966 at MIT and written in LISP-MAD, ELIZA utilizes a knowledge database to identify keywords with the highest rank when responding to a user's query. It then applies transformation rules to sentence patterns provided to the chatbot through scripts. An example of a popular script is called DOCTOR. ELIZA's success inspired researchers to expand on the idea and develop other rule-based chatbot systems.

For a period, there was a notable emphasis on rule-based chatbot programs among researchers, particularly following the development of ELIZA. Subsequently, several similar systems like **PARRY, ALICE, CLEVER, SHRDLU**, and others emerged in the ensuing years. These early rule-based chatbot systems were characterized by their simplicity and reliance on manually crafted rules and sentence patterns outlined in scripts provided to the system. **Consequently, the knowledge base and functionality of these programs were confined to the predefined rules and information.**

Over the subsequent decades, efforts to enhance language understanding saw advancements with the development of more sophisticated rule-based systems. Despite progress, these systems **grappled with the complexities of human language**, struggling to capture contextual subtleties and adapt to diverse linguistic expressions. Researchers increasingly recognized the necessity of transitioning from rigid rule-based

approaches to models capable of learning and generalizing from data. This shift gave rise to statistical language processing techniques like n-grams and hidden Markov models, opening the door to more nuanced language analysis.

This distinction also highlights a significant difference between early rule-based chatbot programs and contemporary advanced models based on neural networks and large language models (LLMs). Another category of early language models relied on information retrieval (IR) techniques. In this approach, chatbots generated responses by matching patterns in pre-constructed databases of conversation pairs. Information retrieval–based chatbots, exemplified by CleverBot from the late 1980s, benefited from techniques like term frequency-inverse document frequency (TF-IDF), cosine similarity, and state space models at the word level.

Statistical Language Processing

While some early designs and models in the field of language modeling have become obsolete in today's advanced architectures, certain fundamental techniques have become integral components of contemporary language models and broader machine learning approaches. Key building blocks for natural language processing include n-grams, bag-of-words (BOW), and TF-IDF.

N-grams

The introduction of n-gram models in the 1990s and early 2000s marked a pivotal advancement in statistical language modeling. Founded on a simple yet potent concept, these models evaluated the likelihood of a word's occurrence by examining the **preceding words within a sequence**.

Despite their uncomplicated nature, n-gram models presented a crucial mechanism for grasping context in language. They represent sequences of n adjacent items from a sample of data, such as words in a sentence. By concentrating on the local relationships between words, these models began capturing the inherent dependencies that shaped meaningful linguistic expressions.

Despite their simplicity, n-grams have played a crucial role in **predicting the probability of the nth word based on the previous n-1 words**. This concept has been instrumental in both basic language modeling and the development of **more sophisticated models**.

Note Google PageRank Algorithm

A noteworthy application of n-grams emerged with Google's groundbreaking PageRank[5] algorithm in 1996. This algorithm transformed web search by employing n-gram analysis to evaluate word co-occurrences across web pages, effectively ranking their relevance. This innovative application showcased the versatility of n-gram models in real-world scenarios extending beyond language processing alone. N-gram models not only underscored the importance of contextual information in language but also paved the way for the development of more intricate techniques capable of capturing a broader spectrum of linguistic nuances.

Bag-of-Words (BOW)

Another fundamental technique in language modeling is the bag-of-words (BOW). This simple approach represents elements of language as numerical values based on word frequency in a document. Essentially, BOW[6] utilizes word frequency to create fixed-length vectors for document representation.

Although straightforward, BOW has been a foundational vectorization or embedding technique. Modern language models often build upon advanced word embedding and tokenization techniques.

The BOW model represents a method for converting a document into numerical form, a prerequisite before employing it in a machine learning algorithm. In any natural language processing task, this initial conversion is essential as machine learning algorithms cannot operate on raw text; thus, we must transform the text into a numerical representation, a process known as **text embedding**.

Text embedding involves two primary approaches: word vectors and document vectors. In the word vectors approach, each word in the text is represented as a vector (a sequence of numbers), and the entire document is then converted into a sequence of these word vectors. Conversely, document vectors embed the entire document as a single vector, simplifying the process compared to individual word embedding.

[5] Lawrence Page, S. Brin, The PageRank Citation Ranking: Bringing Order to the Web, 1996

[6] Z.S. Harris, Distributional structure, Word, vol. 10, no. 2–3, pp. 146–162, 1954

Moreover, it ensures that all documents are embedded in the same size, a convenience for machine learning algorithms that often require a fixed-size input. For instance, with a vocabulary of 1000 words, the document is expressed as a 1000-dimensional vector, where each entry signifies the frequency of the corresponding vocabulary word in the document.

While this technique may be limited for complex tasks, it serves well for simpler classification problems. Its simplicity and ease of use make it an appealing choice for embedding a set of documents and applying various machine learning algorithms. The BOW model offers easy implementation and swift execution.

Unlike other embedding methods that often demand specialized domain knowledge or extensive pre-training, this approach avoids such complexities or even manual feature engineering. It essentially works out of the box. However, its efficacy is limited to relatively simple tasks that **do not rely on understanding the contextual nuances of words**.

The typical application of the bag-of-words model is in embedding documents for classifier training. Classification tasks involve categorizing documents into multiple types, and the model's features are particularly effective for tasks like **spam filtering, sentiment analysis, and language identification**.

For instance, spam emails can be identified based on the frequency of key phrases like "act now" and "urgent reply," while sentiment analysis can discern positive or negative tones using terms like "boring" and "awful" vs. "beautiful" and "spectacular." Additionally, language identification becomes straightforward when examining the vocabulary.

Once documents are embedded, they can be fed into a classification algorithm. Common choices include the naive Bayes classifier, logistic regression, or decision trees/random forests – options that are relatively straightforward to implement and understand compared to more complex neural network solutions.

TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF[7] is yet another statistical measure that leverages word frequency to evaluate the relevance of a word in a given set of documents. In comparison to bag-of-words, TF-IDF is more advanced as it employs two distinct metrics to quantify the relationship between words and documents in a more precise manner.

[7] H.P. Luhn, Statistical approach to mechanized encoding and searching of literary information, IBM Journal of Research and Development, vol. 1,no. 4, pp. 309–317, 1957

TF-IDF finds applications in information retrieval and text mining, particularly in tasks like searching for keywords in documents. It is also a valuable tool in natural language processing applications and language models. Term frequency-inverse document frequency (TF-IDF) also stands as a widely employed statistical method in gauging the significance of a term within a document relative to an entire document collection, known as a corpus.

In the process of text vectorization, words within a text document are converted into importance scores, and TF-IDF represents one of the most prevalent scoring schemes. Essentially, TF-IDF scores a word by multiplying its term frequency (TF) with the inverse document frequency (IDF).

Term frequency (TF) measures the frequency of a term within a document relative to the total number of words in that document.

$$TF = \frac{\text{number of times the term appears in the document}}{\text{total number of terms in the document}}$$

On the other hand, inverse document frequency (IDF) assesses the proportion of documents in the corpus that contain the term. Terms unique to a small percentage of documents, such as technical jargon terms, receive higher IDF values compared to common words found across all documents, like "a," "the," and "and."

$$IDF = log\left(\frac{\text{number of the documents in the corpus}}{\text{number of documents in the corpus contain the term}}\right)$$

The TF-IDF score for a term is determined by the multiplication of its TF and IDF scores. In simpler terms, a term holds high importance when it appears frequently in a specific document but infrequently across others. This balance between commonality within a document, measured by TF, and rarity between documents, measured by IDF, results in the TF-IDF score, indicating the term's significance for a document in the corpus.

$$TF\text{-}IDF = TF * IDF$$

TF-IDF finds applications in various natural language processing tasks, including search engines, where it is used to rank document relevance for a query. It is also applied in text classification, text summarization, and topic modeling.

It is important to note that different approaches exist for calculating the IDF score. The calculation often involves using the base 10 logarithm, although some libraries may opt for a natural logarithm. Additionally, adding one to the denominator is a common practice to prevent division by zero.

Vector Space Models and State Space Models

Notably, the bag-of-words (BOW) model is considered a **basic word embedding technique**, correlating words to vectors. However, the widespread adoption of embedding words in vector spaces saw significant advancements, particularly with breakthroughs in the early 2000s and later with **Word2Vec**[8] **and GloVe in 2013 and 2014**, respectively. Word embedding techniques have become integral to language modeling, extensively employed by various other models.

Vector space models and state space models are fundamental concepts in language modeling. Vector space models involve **algebraic approaches** for representing language elements as vectors embedded in specific vector spaces.

A vector space comprises vectors, numerical representations of words, sentences, and even documents. While basic vectors, such as map coordinates, have only two dimensions, those employed in natural language processing can encompass thousands.

The concept of representing words as vectors, often known as **word embedding or word vectorization**, has been present since the 1980s.[9] This simplifies the assessment of similarity between words or the relevance of a search query to a document. Cosine similarity is commonly applied to gauge the similarity between vectors. Utilizing linear algebra with nonbinary term weights, the vector space model enables the computation of the continuous degree of similarity between two objects, such as a query and documents, facilitating partial matching.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, Efficient estimation of word representations in vector space, preprint arXiv:1301.3781,2013

[9] D.E. Rumelhart, G.E. Hinton, R.J. Williams, et al., Learning internal representations by error propagation, 1985