2
An Overview of Natural Language Processing and Neural Networks
Learning Objectives
After completing this chapter, the readers are expected reLearn the basics of Natural Language Processing ALP necessary for understanding the subsequent chapters.
Gain knowledge about various semantic and syntactic paradigms in NLP
Learn the basics of Neural Networks required to understand the subsequent chapters.
Become familiar with the evaluation metrics employed in neural network modeling dfgjskdhf
The natural evolution of languages has enabled humans to communicate and share ideas across geopolitical boundaries. However, the concept of a machine processing and understanding these human languages is a complex challenge. Since machines inherently understand numbers and numerical operations, it is necessary to convert natural language into a format that computers can comprehend. This is where ALP comes into play. ALP is an interdisciplinary field within computer science that encompasses techniques to make human language accessible and interpretable by machines The foundational concepts in ALP draw from a wide range of disciplines, including theoretical computer science linguistics, statistics, and artificial intelligence.
Statistical methods and linear machine learning models have long been utilized in ALP tasks. However, these traditional approaches depend heavily on manually crafted dictionaries and features, limiting their capacity to capture complex patterns and semantics in language As the volume of textual data has increased and computational resources have become more accessible neural architectures have gained prominence in modern ALP techniques These neural models excel at capturing latent knowledge from data processing inputs of varying lengths, leveraging information from long sequences of text, and autonomously learning features, reducing the need for extensive manual efforts.
This chapter, is divided into two parts. In Part I, readers are first introduced to the fields of linguistics, and ALP Section 2.1 discusses the goals of computational linguistics, and ALP Section 2.2 describes various tasks. in ALP and introduces the ALP pipeline. Section 2.3 explores the linguistic components of language such as morphology, lexicon, and text, normalization techniques like stemming and lemmatisation. In Section 2.4, we provide an overview of different tokenisation and semantic analysis techniques Section 2.5 focuses on syntax and grammar-based parsing methods while Section 2.6 delves into semantics and semantic parsing Finally, Section 2.7 presents the task of language modeling emphasizing conditional probability and the frequency of co-occurrence.
In Part II, we explore neural networks and related concepts to set the foundation for deep learning techniques discussed in later chapters. Section 2.8 introduces the perception and its applications in modeling a linear classifier. Section 2.9 presents multiplayer perceptions and popular non-linear activation functions. Section 2.10 covers the gradient-based training process for neural networks and error backpropagation. In Subsection 2.10.3, we discuss the various hyperparameters that influence neural network training Section 2.11 focuses on the challenges that affect gradient descent. Finally, in Section 2.12, we examine the performance measures commonly used to evaluate deep learning tasks.
Part I, Natural Language Processing Since the advent of computers researchers have been captivated by the idea of teaching machines to interact like humans As early as 1963, Joseph Weizenbaum developed ELIZA Weizenbaum 1983), a released sabot designed to converse with humans Fast forward to 2014, Eugene Dustman a sabot passed the Turing Test, with human judges unable to discern that Eugene was, in fact, a bot. In 2017, Google revolutionized the field by introducing a new machine translation architecture, now famously known as Transformers Aswan er all 2017). More recently, the success and widespread adoption of language models like CatGUT and its successors have generated immense interest in language models accelerating research in both ALP and deep learning This book is motivated by the need to enhance our understanding of language models and to demystify the concepts from beginner to advanced levels.
Computation aside, a very nuanced problem with language is its ambiguity, contextualization and its dynamic nature relative to the zeitgeist. Consider the sentence I, saw her duck’. One interpretation could be that I, saw a duck’. (noun) that belonged to her Another interpretation of this same sentence could be that I, saw her perform the act of ducking (verb) to avoid an obstacle. Depending on the context, sentences with similar structures and subjects can still convey very different meanings. Take the following sentences I, ate rice with a spoon’, I, ate rice with currant I, ate rice with Raoul All of these sentences convey that I, am eating rice In the first instance, I, use a spoon’, as a utensil, while in the second sentence I, use curd as an accompaniment. The third sentence however, suggests that I, am eating rice in the presence of a person. Teaching computers to understand how to interpret these sentences differently and resolve ambiguities has been a significant motivation behind the development of ALP and understanding techniques Computational Linguistics and Natural Language Processin
Like any other medium of information human languages have undergone various stages of development—they originated at some point in time, propagated far and wide and have borne witness to the evolution of human society. While we do not know exactly when and how the earliest humans spoke, numerous theories have been proposed about the origin of language Blister Hardy and Elaine Morgan proposed the aquatic ape theory in 1997, highlighting that there are certain traits we do not share with our primate relatives but that we do share communication traits with aquatic animals. For example, whales and dolphins are known to communicate with members of their species using sounds.
Linguistics is the discipline that engages in the scientific study of languages It is an interdisciplinary system (see Figure 2.1) where linguistics, and its branches, such as sociolinguistics, psycholinguistics, and neurolinguistics, seek to answer significant philosophical questions, such as What rules do languages follow? How do languages evolve? How do we learn and process meanings. in our minds? How are different modalities of languages related to each other Linguists like Foam Chomsky and Steven Pinker hypothesis that language is something innate. From a very early age, we begin to mimic natural sounds and lip movements. We grow up associating these sounds with certain objects, qualities, and attributes of the environment around us. By studying these theories we can infer what the earliest languages might have been like but our understanding only extends so far back into the history of human languages (Jurafsky and Martin 2009; Pinker 2010; Vaneechoutte 2014).
[Insert Figure Figure 2.1: Language-related Disciplines (Tsujii 2021) – Linguistics Cognitive Science, Psychology, Natural Language Processing ALP Artificial Intelligence (AI), and Computational Linguistics All of these disciplines, study language from different perspectives.
Many other fields such as neuroscience and psychology, also show great interest in languages Computational linguistics, is a outfield of both linguistics, and computer science that focuses on the interactions between human language and computers serving as a bridge between the broader field of linguistics, and engineering processes. While computational linguistics, is more concerned with understanding language structure and developing computational models ALP emphasis's the design and analysis of algorithms and systems for tasks. that rely on processing human language input.
2.2 Overview of the Natural Language Processing Pipeline
The standard pipeline. used in ALP involves several steps. The natural language input.
2.2 for the pipeline. discussed in this book is typically modeled as a collection of machine-readable text, documents, known as a corpus; a larger collection of these documents, is referred to as corpora. When employing ALP systems the standard pipeline. consists of a sequence of steps. as illustrated in Figure 2.2, to address one or more NLP-based tasks.
[Insert Figure]
Figure 2.2: Stages of the language processing pipeline. for textual data input.
Tasks in Natural Language Processing ALP Depending upon the task the output from the ALP pipeline. could be in the form of a sentence-level or word-level class label, a sequence of words, a piece of text, and even paths of a graph node-to-edge sequence (see Figure 2.3). To better illustrate the tasks. we will consider an example, I, do not support WHO. They underfed Indian diseases’. Below is a non-exhaustive list of popular tasks. in NLP.
Sentiment Analysis: Detecting the type and intensity of emotional tone or opinion expressed in some text, Here, the output fr the entire sentence is a label, such as positive, negative or neutral. Based on the usage of the phrase do not support we can label, the sentiment of the above example, as ‘Negative’.
Part-of-Speech POSS Tagging: Figuring out the grammatical class (noun) pronoun, adjective, adverb, etc.) of each word in a sequence The output is a sequence of class labels tagged for each word in the sentence The above example, can be tagged as I, pronoun, do (verb) not adverb, support (verb) WHO. (noun) (punctuation). They pronoun, underfed (verb) Indian adjective, diseases’. (plural (noun) . (punctuation). Note that in ALP we use a diverse set of POSS tags. For instance, the Penn Treebank project uses 36 POSS tags.
Named Entity Recognition HER Identifying and classifying (noun) phrases into dreamworld entities like organization country, groups, nationality, etc.) Here, the output is a label, for one or more contiguous words, In our example, the terms WHO. and Indian will be tagged as ‘ORG’ for organization and CORP for nationality, respectively.
Text Entailment: Determining whether the premise sentence implies, contradicts, or has nothing to do with the preceding hypothesis sentence The output is a label, for the sentence pair. In our example, the premise of ‘underfunding’ supports the hypothesis of not supporting’.
Semantic Role Labeling Identifying the role of each (noun) phrase with respect to the predicate of the sentence The output is a label, for a phrase.
Machine Translation: Conversion of text, from one human language to another. Our example, when translated into Hindi, will be मैं WHO. का समर्थन नहीं करता. वे भारतीय बीमािरयों के िलए कम फंड देते 2014).
[Insert Figure]
Figure 2.3: Tasks in ALP Summarisation: Producing a shorter version of the larger reference text, that still retains the information conveyed. The output is a piece of text, that is way shorter in length than the reference Question Answering: Providing the correct and concise answer to a user query. The output is a piece of text, that mimics human response. Based on our example, sentence we can ask the system ‘Which country, does WHO. underfed and expect the answer to be ‘India’.
9. Knowledge Graph Completion: Filling missing information in a structured knowledge graph by using world knowledge The output is in the form of an edge that is not in the edge list of a graph For example, an edge to store or predict can be .
Data Acquisition. To enable a machine to learn from data the primary requirement is the availability of that data often obtained through a combination of duration strategies such as web scraping, synthetic data generation, and manual annotation. In most cases, the goal is to gather a large collection of unstructured, free-flowing text, fragments or documents, which may or may not be annotated by a human expert. When machine-readable text, is unavailable, such as when scanning text, from PDF Optical Character Recognition (OCR) proves useful. Public datasets and text, dumps are typically the first sources to explore for open-domain text, documents.
Data Cleaning. Since most textual data is curated from the web it typically requires cleaning before further processing This is because the data might contain markup, special characters, personal information poorly formatted tags. and other unwanted elements. By employing regular expressions, handling stray characters, and using dictionaries to correct misspell words, we can effectively reduce noise and perform reduplication Additionally, the data might be encoded in different Unicode formats, so appropriate logic must be applied to address such encoding issues.
Pre-processing. This step involves breaking the text, into smaller units and then normalizing it using techniques such as lowercasing, shopworn removal, stemming and lemmatisation. which will be discussed in this chapter, In certain contexts, digits and (punctuation). may also be removed if they contribute little to the overall information It is crucial to note that there is no one-size-fits-all reprocessing technique applicable to all ALP tasks.
Feature Engineering. Once the text, has been reprocessed we now need to represent the text, in a way that a machine can understand As machines reduce everything into numbers we build a text, representation by encoding it into a numeric vector. In ALP or deep learning encoding can be considered as a mapping function that takes input.
2.2 in raw human-readable form text, images, videos) and converts it into numerical vectors for computational methods to be applied to them. However, there can be multiple ways of performing encoding depending on the task the datasets and the computational resources available at hand. This is where feature engineering helps. It helps. us. analyst the essential features, and most informative parts. of the input.
2.2 and only use those to encode the input.
2.2 so that we can encode maximum information in as little memory as possible. Encoding can be achieved by simple frequency-based heuristics such as ocelot encoding and bag-of-words representation ALP practitioners these days use probabilistic, neural approaches to learn word embedding which are representations of words, in the feature space. Parallel to encoding decoding is a map function for converting numerical vectors into human-readable symbols (texts, pixels, etc.).
Model Building. Once we have encoded the textual data it can be passed to any machine learning or deep learning model to learn from the corpus; Irrespective of whether the task is one of BLU or LG learning from textual data boils down to sequence modeling since text, can always be represented as a sequence of words/phrases/characters. As NLP-based sequence models aim to learn the ‘hidden/latent language in the vector. space. from the input.
2.2 text, they are also called Language Models LBs For a long time, neural networks like Recurrent Neural Networks RN Elan 1990), Long Short-Term Memory ATM and Gated Recurrent Units GPU Gees er all 2000; (Tsujii 2021; Che er all 2014) showed incredible performance in modeling sequential data like text, However, with the introduction of transformers in 2017 Aswan er all 2017), there has been a significant leap in the length and complexity of the textual data that can be modeled with transformers becoming the dd factor standard in today ALP In the coming chapters. we will introduce different language models along with the fundamental concepts on which LBs are built.
Evaluation. Once the model is designed we must assess how ‘good’ a language model is While classification tasks. can be evaluated using existing accuracy and Score (macro/micro), newer metrics need to be devised for tasks. that involve generating text, For machine translation and summarisation tasks. we typically use Bilingual Evaluation Understudy BLEW and Recall-Oriented Understudy for Gusting Evaluation (ROUGE) scores, which capture the lexical and syntactic overlap between the expected and predicted text, Meanwhile, newer semantic measures like BERTScore have also been designed When comparing two LBs themselves, we can employ entropy-based measures like perplexity. We introduce these KM evaluation metrics in Chapter 4.
Information
Meaning
Phonetic
How a word is expressed vocally with a certain sound.
Structural
How a word is composed of different linguistic components.
Syntactic
How a word fits into the overall structure of a sentence
Semantic
What is the meaning of a word in some particular context?
Pragmatic
How a word is used in a discourse or conversation.
Table 2.1: Different kinds of information that can be derived from a word.
Deployment. Transforming a trained model into a functional component of a software system in any neural pipeline. requires exporting the model and specifying the environment (libraries and versions), hyperparameters for the model and the model itself. Language models are often published on open-source forums like the Hugging Face platform. When a model has to be made available as a service for inference, robust monitoring also needs to be set to ensure performance and model safety at scale, apart from an optimized input.
2.2 reprocessing pipeline. as discussed above In many production systems a feedback loop is also implemented to improve the model over time, Morphology
Different forms of information involved in the processing language are listed in Table 2.1. In this chapter, we will focus on structural, syntactic and semantic information processing.
Before we delve into computational methods of developing language models it is imperative to examine the most fundamental units of linguistic structure – the word Words play. an integral role in our ability to use language to express our emotions and creativity, originating from the fundamental question: what do we know when we know a wordAge formal study of the internal structure of words, and the relationship among words, is called morphology, The term itself. is derived from the Greek word morphs meaning form and logy meaning the branch of knowledge Morphology also refers to our internal grammatical knowledge concerning the words, and how their usage change based on language geography, context, and time, Languages like Hindi, Turkish, and Hungarian are considered morphologically rich, whereas English and Chinese are morphologically poor. In morphologically rich, languages the word forms of some word classes, like verbs, may vary a lot depending on the context, Take the phrase will go’ with its usage as described in Table 2.2.
English
Hindi
Tamil
I will go’ जाऊंगा
நான் We will go’ जायेंगे
நாம் ேபாேவாம
You will go’ जाओगे
நீ ேபாவாய
He will go’ जाएगा
அவன் ேபாவான
She will go’ जाएगी
அவள் Table 2.2: Different forms of the token will go’ in morphologically-poor English and morphologically-rich languages Hindi, and Tamil). Morphologically-rich languages have various forms to represent the same token depending upon the subject in the sentence Such languages also have additional grammatical classes.
For morphologically-poor English irrespective of whether the action is being performed by a single person. a group of people, or by people, of different genders, the phrasing will go’ remains the same Meanwhile, in a morphologically-rich language like Hindi, the phrasing will get modified to suit the respective form depending on the preceding (noun) form (plurality, gender, etc.) and tense form first person. third person. etc.) Morphemes
Words are composed of atomic building blocks called morphemes. The words, ‘taking’ and ‘courses’, for instance, are made up of basic units like take and course, and the other blocks like inf and -s convey additional meanings. such as a sense of the nature of action or (plurality, respectively. Some morphemes. independently constitute a word by themselves, They are called free morphemes. The word fish, for example, consists of a singular free morpheme, the word itself. with a predefined meaning Other morphemes. are not words, by themselves, but are parts. of words—these are bound morphemes. Affixes are the most common type of bound morphemes. They attach to a base word or a stem and modify its meaning in some way or another. For example, the word ‘taking’ consists of the suffix morpheme, inf attached to the base ‘take’.
Nouns, verbs, adjectives, and adverbs are put into the bucket of content words, Content words, are often called open class words, because we regularly add new words, to this bucket Other classes, of words, do not have precise lexical meanings. or obvious concepts associated with them. including conjunctions and or prepositions to from at with articles a an the quantifiers all few, many some demonstratives this that and pronouns. These kinds of words, are called function words, because they serve a grammatical function They are also called closed class words, as most languages have a small, fixed number of words, that fall into this bucketS root is the base form of a word that cannot be analyses or reduced further without destroying its meaning For example, in terms of conserving its meaning the term ‘forest’ cannot be broken down into for and ‘est’. Complex words, may consist of a morpheme, root and one or more affixes. Affixes like en dis, miss re, non, sub, super, anti, inter, and infra that are attached to the beginning of another. morpheme, are called prefixes. Similarly, suffixes are morphemes. that get added at the end, such as inf mess la and able. A morpheme, that is a prefix in one language in a semantic sense maybe a suffix in another. and vice versa.
Original Word
Stemming
Lemmatisation
witnessed
wit
witness
assignments
assign
assignment
considerable
consider
considerable
democratisation
democrat
democratisation
interpolated
interpol
interpolate
effectively
effect
effective
Table 2.3: Comparing the results of Porter Stemmed and WordNetLemmatizer algorithms for various words.
Historically, we have been following morphological rules that govern how these affixes. attach to the base word For instance, when we add prefixes. the resulting word is formed by putting together the two morphemes. asks egg ere + flight = relight In contrast, the resulting word might not be a simple concatenation in many suffixes egg ready + la = readily). In English as well as many other languages apart from attaching affixes. new words, can also be formed by compounding existing words, where individual words, like ‘black’ and ‘board’, can be joined together to form a compound word like ‘blackboard’. In other cases, words, like will and ‘would’ are contracted to -’ll and -’d and attached to the end, of words, Identifying the various parts. of a word into the morphemes. that it is composed of and producing its structured representation is called morphological parsing or stemming Stemming stemming algorithm or stemmed is the one that eliminates affixes. and serves as a heuristic to normalize the inflectional (plurals, tenses, etc.) and derivational (turning verbs, into nouns) forms of a word For example, the words, run, runs, ran, and running all refer to the same underlying concept and can be represented by a single concept instead of four different ones. However, stemming can be tricky as we can lose information by chopping off a few, characters, of a word indiscriminately. In order to support stemming a variety of heuristics released algorithms have been proposed ALP packages often include the famous stemming algorithms—the Porter and Snowball Stemmers.
A stem may not be a valid dictionary word but merely an abstraction that represents all the words, that look the same at the character level. For instance, if we have a stemming rule to remove all instances of -s from the end, of words, in order to normalize (plural forms we might end, up with non-meaningful results as well – ‘lens’ becomes ken which is not a known English dictionary term yet will be acceptable as per the stemming rules Lemmatisation
Instead of normalizing the words, at the superficial character level. we can group them. based on their larger context, and usage Lemmatisers are algorithms that normalize words, down to the underlying semantic form – the lemma. Lemmatisers are usually more accurate than stammers as they use a knowledge base or thesaurus of words, their synonyms, and forms to ensure that only words, that mean the same are clustered together and are represented by a well-defined lemma. instead of an arbitrary stem which may not be a dictionary word This difference is easier to understand with the examples in Table 2.3. A lemmatiser will be able. to group the words, ‘good’ better and ‘best’ into the same bucket if it knows that these words, are adjectives, A table or dictionary lookup is often the way how lemmatisers retrieve information about similar-meaning words, WordIer is a famous database of English words, that are linked together by semantic relations.
2.3.4 Lexicon
Stemming or lemmatisation. helps. reduce the signal-to-noise ratio in a text, corpus; by reducing the redundant concepts present in it The process allows us. to build an optimal vocabulary/lexicon that makes up the language of the corpus; This lexicon, defines the input.
2.2 and output space. for the language model trained on the corpus; Many classical tasks. in ALP like sentiment analysis HER and POSS tagging, as well as domain-specific tasks. like medical or legal text, analysis depend upon a lexicon, for making sense of the input.
2.2 For many of these tasks. we prefer to use specialized lexicons egg FINN SentiWordNet, EmbLem PropBank) that are built up by manually annotating with the help of human experts, automatic extraction using statistical and machine learning techniques or using a hybrid approach. The intuition behind the lexicon, also plays a role in the formation of rules and conventions to incorporate new terms like ‘tweet’ and angry They can be formed due to the adoption of popular culture, foreign words, compounding or due to morphological changes.
2.4 Tokenisation
In order to build the lexicon, the question: is how we define the boundary of breaking the text, stream into entities that can be added to the lexicon, Commonly, these informative units of information in ALP are called tokens, and the process of obtaining tokens, by breaking the text, corpus; into smaller processable units/chunks is called tokenisation.
For example, consider an input.
2.2 corpus; consisting of two sentences S1: I, want the first token from the list of tokens, and S1: The tokens, are obtained via tokenisation’.
Sentence/Word/Character-Level Tokens. For the above example, sentence-level tokenisation will yield a list I, want the first token from the list of tokens, The tokens, are obtained via tokenisation by splitting the sentences at the (punctuation). marks. However, this naive splitting at (punctuation). can also be problematic; the phrase ‘But, here we are can be wrongly split at ‘,’ instead of ‘!’. To reduce ambiguity, one can also split the sentence at the word level. In English word level. splitting is easier as whitespace is the default delimiter. Therefore, our text, ‘would’ be broken as the following word tokens, I, want the first token from the list of tokens, The tokens, are obtained via tokenisation When splitting by spaces, the tokens, tokens, and tokenisation have (punctuation). attached to them. To reduce the number of unique tokens, in tokenised output we can either discard the (punctuation). altogether or add them. separately to the token list as I, want the first token from the list of tokens, The tokens, are obtained via tokenisation ‘.’]. Once we have obtained the words, we can make the chunking process even more granular by operating at the character level. In that case, our corpus; will be listed as I, ‘ ’, ‘w’, a ‘n’, ‘t’, ‘t’, ‘h’, ‘e’, ‘f’, ‘i’, ‘r’, -s ‘t’, ‘t’, ‘o’, ‘k’, ‘e’, ‘n’, ‘f’, ‘r’, ‘o’, ‘m’, ‘T’, ‘h’, ‘e’, ‘l’, ‘i’, -s ‘t’, ‘o’, ‘f’, ‘t’, ‘o’, ‘k’, ‘e’, ‘n’, -s ‘t’, ‘h’, ‘e’, ‘t’, ‘o’, ‘k’, ‘e’, ‘n’, -s a ‘r’, ‘e’, ‘o’, ‘b’, ‘t’, a ‘i’, ‘n’, ‘e’, -’d ‘v’, ‘i’, a ‘t’, ‘o’, ‘k’, ‘e’, ‘n’, ‘i’, ‘z’, a ‘t’, ‘i’, ‘o’, ‘n’, In this example, of character-level chunking it becomes difficult to detect word boundaries. In later paragraphs, we will discuss how to overcome this issue.
N-grams. So far we have observed tokens, as one unit at a time, This form of token is also called engram with uni being the unit of tokenisation referencing the quantity one However, we can also look at neighboring tokens, such as ‘n’, tokens, ahead of the current token leading to grams instead For example, when ‘n’, = 2, our word-level tokens, will be of the form I, want want the the first ..., via tokenisation tokenisation DOS where DOS is the unique token indicating we have reached the end, of our text, stream Similarly, for ‘n’, = 3, the first token will be I, want the and so on The gram operation can be performed at the sentence or character level. as well As the window size for ‘n’, increases, we are able. to capture more semantic context, however, with a very large value of ‘n’, we end, with the whole stream defeating the purpose of performing chunking The task of obtaining the optimal number of grams is task and data-specific.
2.4.1 Advanced Techniques: Sword Tokenisation
On the one hand. character-level tokens, provide more resilience against spelling errors. On the other hand. it comes at the cost of semantic information For example, the sword ‘ken’ can be part of semantically diverse terms ‘Kendall’, token or broken Here, practitioners have come up with a tokenisation process that is a combination of word and character levels tokens, known as sword tokenisation which is primarily based on splitting and merging tokens, based on the frequency of occurrence within a corpus; In this section, we discuss the two most widely adopted bottomed sword tokenisation techniques that take a greedy approach. based on the frequency of sword occurrence—Byte Pair Encoding and Codpiece Tokenisation.
<H4> Byte Pair Encoding (BPE)
Byte pair. encoding (Gage 1994) was, initially developed as an algorithm to encode/compress a text, based on the most frequently occurring bytes a byte or 8 bits refers to a single character token for practical usage The algorithm merges the most frequently occurring consecutive bytes and replaces them. with a new representative token that is not part of the existing lexicon, The process continues until no more merger is possible. (see Algorithm In order to preserve word boundaries. the space. token is replaced by a special token say ‘w’, which is not a part of the vocabulary and is concatenated to the last character of each word Thus, our yd and 3rd words, in the corpus; ‘w’, a ‘n’, ‘t’, ‘t’, ‘h’, ‘e’, will be represented as ‘w’, a ‘n’, ‘t’, ‘w’, ‘t’, ‘h’, ‘e’, </w>’].
FCBPE ‘i’, j) = ‘i’, : example 2.1. Taking our initial corpus; into consideration, let us. observe a few, iterations of REIteration 0 (pre-tokenisation): Our tokens, are enlisted as ‘i’, ‘w’, a ‘n’, ‘t’, ‘t’, ‘h’, ‘e’, ‘f’, ‘i’, ‘r’, -s ‘t’, ‘t’, ‘o’, ‘k’, ‘e’, ‘n’, ‘f’, ‘r’, ‘o’, ‘m’, ‘t’, ‘h’, ‘e’, ‘l’, ‘i’, -s ‘t’, ‘o’, ‘f’, ‘t’, ‘o’, ‘k’, ‘e’, ‘n’, -s ‘t’, ‘h’, ‘e’, ‘t’, ‘o’, ‘k’, ‘e’, ‘n’, -s a ‘r’, ‘e’, ‘o’, ‘b’, ‘t’, a ‘i’, ‘n’, ‘e’, -’d ‘v’, ‘i’, a ‘t’, ‘o’, ‘k’, ‘e’, ‘n’, ‘i’, ‘z’, a ‘t’, ‘i’, ‘o’, ‘n’, Our unique vocabulary is enlisted as ‘i’, ‘w’, a ‘n’, ‘t’, ‘h’, ‘e’, ‘f’, ‘r’, -s ‘o’, ‘k’, ‘m’, ‘l’, ‘b’, -’d ‘v’, iteration 1: Among the possible. character combinations, the most frequently occurring character pairs are ‘o’, + ‘k’, occurring 4 times. Thus, all occurrences of ‘o’, + ‘k’, will be replaced by oi The updated tokens, thus appear as ‘i’, ‘w’, a ‘n’, ‘t’, ‘t’, ‘h’, ‘e’, ‘f’, ‘i’, ‘r’, -s ‘t’, ‘t’, oi ‘e’, ‘n’, ‘f’, ‘r’, ‘o’, ‘m’, ‘t’, ‘h’, ‘e’, ‘l’, ‘i’, -s ‘t’, ‘o’, ‘f’, ‘t’, oi ‘e’, ‘n’, -s ‘t’, ‘h’, ‘e’, ‘t’, oi ‘e’, ‘n’, -s a ‘r’, ‘e’, ‘o’, ‘b’, ‘t’, a ‘i’, ‘n’, ‘e’, -’d ‘v’, ‘i’, a ‘t’, oi ‘e’, ‘n’, ‘i’, ‘z’, a ‘t’, ‘i’, ‘o’, ‘n’, with oi added to the unique count.
Iteration 2: Now, looking at all paired frequencies with oi considered as a single unit we observe that ‘t’, + oi occurring four times. is the next pair. to be merged. This updates the word list as ‘i’, ‘w’, a ‘n’, ‘t’, ‘t’, ‘h’, ‘e’, ‘f’, ‘i’, ‘r’, -s ‘t’, took ‘e’, ‘n’, ‘f’, ‘r’, ‘o’, ‘m’, ‘t’, ‘h’, ‘e’, ‘l’, ‘i’, -s ‘t’, ‘o’, ‘f’, took ‘e’, ‘n’, -s ‘t’, ‘h’, ‘e’, took ‘e’, ‘n’, -s a ‘r’, ‘e’, ‘o’, ‘b’, ‘t’, a ‘i’, ‘n’, ‘e’, -’d ‘v’, ‘i’, a took ‘e’, ‘n’, ‘i’, ‘z’, a ‘t’, ‘i’, ‘o’, ‘n’, with took added as a vocabulary term.
Iteration N: After N: merger and replacement steps. our words, will be represented as ‘i’, ‘w’, a ‘n’, ‘t’, the ‘f’, ‘i’, ‘r’, st’, token ‘f’, ‘r’, ‘o’, ‘m’, the ‘l’, ‘i’, st’, ‘o’, ‘f’, tokens, the tokens, a ‘r’, ‘e’, ‘o’, ‘b’, ‘t’, a ‘i’, ‘n’, ‘e’, -’d ‘v’, ‘i’, a token ‘i’, ‘z’, a ‘t’, ‘i’, ‘o’, ‘n’, and our final vocabulary will be ‘i’, ‘w’, a ‘n’, ‘t’, ‘h’, ‘e’, ‘f’, ‘r’, -s ‘o’, ‘k’, ‘m’, ‘l’, ‘b’, -’d ‘v’, ‘z’, oi took ‘en’, token eh the st’, ‘tokens’]
Once the vocabulary is learned from the initial corpus; the algorithm can break any word it has seen in the corpus; or not seen before an on-the-fly word based on the sword token it has learned For example, the new word ‘mist’ will be tokenised into ‘m’, ‘i’, st’, with ‘m’, ‘i’, and st’, forming the swords Including the word boundary we can represent ‘mist’ as ‘m’, ‘i’, st’, ‘w’, Note that the swords do not have to be actual dictionary terms with a meaning attached to them.
Algorithm 1 The steps. for sword tokenisation as adopted by BEE and codPiece The actual formula in Step 4 is realized by Equations (2.1) and (2.2), respectively. for BEE and WordPiece.
Algorithm 2 Algorithm for obtaining unique tokens, in the corpus; via splitting at the word level.
Input: Vocabulary size ‘k’, Corpus D, Maximum Iteration maxiter
Output: Vocabulary V
1: V
1: ← PREPROCESS(D)
2: ‘i’, ← 0
3: while V
1: < ‘k’, or ‘i’, < master do fl : tr ← max FCC ‘t’, ‘l’, ‘t’, rte ← fl : tr
6: tl:tr ← eleV ← V
1: ∩ Teri ← ‘i’, + 1
9: end, while
return Input Corpus Output Vocabulary V
1: V
1: ←{}
2: for ‘w’, ∈ splitS delimiter. ") dolor ch ∈ ‘w’, diV ∩ send for end, for
return CH codPiece Tokeniser
The processing of merging characters, in BEE depends solely on the frequency count of the characters, at each iteration. Instead of maximizing information gain purely based on frequency we can maximize the likelihood of improving the swords coverage within the corpus; A famous probabilistic, variant of BEE is the codPiece Tokeniser. Keeping the rest of the BEE process the same the primary modification that codPiece introduces is by replacing the exact frequency count in Algorithm 1 at step 4 with Equation By dividing the frequency count Σ of the reoccurring pair. by the product of individual frequency counts Equation 2.2), codPiece penalizes those pairs that are highly frequent in the corpus; In other words, if particular terms are by themselves, high frequency then their combination provides lesser information gain compared to combining less frequent terms SentencePiece Tokeniser
So far both the tokenisation methods we have examined require the corpus; to be split at a word level. and be reprocessed at an individual word level. for which the assumption is the language of the corpus; contains spaces, as delimiters. However, there are some languages like Chinese or Japanese, in which space. delimitation is not available Thus, a language-agnostic/space-agnostic approach. is required Here, the SentencePiece tokeniser comes into play. SentencePiece incorporates a number of techniques to improve upon the existing tokenisation setup. SentencePiece employs Unicode Normalization to work with raw (texts, It employs heap sort to keep track of the vocabulary size ‘But, most importantly, unlike BEE and codPiece which employ a (pre-tokenisation): step Step 2, Algorithm 1), SentencePiece is capable of working with raw (texts, Syntactics
As per the Oxford Dictionary, the term syntax refers to the rules/grammar that state how words, are placed and used in a language to form sentences The syntax is based on the grouping of words, in a natural order 2
An English sentence is composed of a group of words, that form the Noun Phrases (NP) and the Verb Phrases (VP). For instance, in the sentence The old house in the neighborhood is being demolished’, the (noun) phrases the old house and neighborhood can be combined as a single (noun) phrase The old house in the neighborhood Meanwhile, the phrase is being demolished’, is the (verb) phrase The whole sentence can syntactically be represented as S1: → N: P + V
1: P with N: P further composed of N: P → N: P + N: P This process of mapping words, and groups, of words, phrases into their grammatical units is called syntax parsing While linear representation in terms of rules is a way to decompose the sentence it can also be represented in the form of a hierarchy or a tree, with the words, forming the leave nodes, the grammatical constituents forming the intermediate nodes, and the sentence forming the root node. In this section, we provide an overview of three ways in which the syntax tree, can be parsed. The parse trees act as an abstraction of the sentence.
[Insert Figure]
Figure 2.4: Constituent Parsing for the Sentence, The mouse ate the cheese that was, kept in the drawer’.
Dependency Parsing While performing POSS tagging, and constituency parsing we implicitly looked at the relation among the words, to assign adequate tags. and phrases Still, the information was, insufficient to answer questions, such as What did the mouse eat? or Where was, the cheese kept In such cases, we need to mark the relation between mouseate–cheese-drawer explicitly. Being able. to state the subjects and objects, in a sentence along with the relationship among them. is known as dependency parsing The dependency grammar describes the structure of a sentence in terms of the words, and the grammatical relationship that holds between words, The dependency relations.
2.3.4 thus act as a proxy to the semantic relations.
2.3.4 in text, These binary relations.
2.3.4 consist of a head and a dependent. The head is the central word in a constituent egg (noun) in a (noun) phrase (verb) in a (verb) phrase All other words, are dependent. on the head In a dependency parse tree, the heads are linked to words, that are immediately dependent. on them. The main (verb) of the sentence is the root node. from which one can follow? a unique directed path to each word in the sentence Such a parse tree, is flexible with word order and is helpful in parsing morphologically rich, languages as well Figure 2.5 shows the parse tree, for an example, sentence The mouse ate the cheese that was, kept in the drawer’. The actual parsing is realized through transition-based state spaces, that use stacks to create dependency structures and graph-based methods that use maximum spanning trees.
[Insert Figure]
Figure 2.5: The dependency parse tree, for the sentence The mouse ate the cheese that was, kept in the drawer’. The labels on the arcs are according to Universal Dependency nomenclature for grammatical relations.
2.3.4 Semantics
In the last section, we saw how grammatical abstractions can help answer simple questions, within a sentence Instead of the question: What did the mouse eat? if we were to ask ‘Which furniture is being referred to then the notion of furniture and drawer’. being concepts that are close to each other needs to be established. This idea of establishing closeness of concepts that may linguistically or grammatically not appear close to each other is known as semantic similarity. Semantics, in turn, can be defined as the underlying meaning associated with the entity under consideration, Semantics, help access what is the relation that different words, have with each other when present together in a sentence For example, when presented with a stimulus word ‘bank’, we think of other response. words, like ‘money’, ‘river’, and ‘blood’ depending on the context, in which the stimulus word is used The way a language evolves plays a central role in explaining these relations.
2.3.4 Here, word association can be defined as a relationship between words, in a language based on their meaning Semantics, is not just concerned with the meaning of words, but also how to combine words, into meaningful phrases and sentences For example, the phrases not honest’ and the word ‘dishonest’ carry the same connotation/semantics even though the terms are medically different.
Semantic parsing involves mapping the natural language input.
2.2 to a logical form that connects the language to dreamworld concepts Unlike syntactic parsing methods which focus solely on structure and grammar semantic parsing methods try to extract the meaning and context, of a sentence A semantic parser consists of a formal knowledge representation technique and an inference, mechanism. One of the ways to represent language formally is by translating a sentence to first-order logic where the predicates are the words, in the sentence In order to represent the words, as predicates we need to be sure of the sense in which the word has been used in the sentence Even if we can represent words, and relationships as predicates these do not make sense on their own. 2
An external knowledge source is required to help us. define rules that use these predicates and learn the semantic logic In this section, we briefly describe three techniques for semantic parsing—decomposition, ontology, and distributional statistics.
Decompositional Semantics, We can derive the meaning of a word by dividing it into various semantic components or qualities, For instance, in the sentence The mouse ate the cheese that was, kept in the drawer’. the word mouse implies, that the subject of the sentence is a mammal and a terrestrial but not a human However, if the word mouse was, replaced with the word ‘boy’, it ‘would’ imply that the subject has all of the three qualities—being a mammal a terrestrial and a human These decomposed semantics can also be mapped to first-order logic such as mouse ⇒ mammal ∧ terrestrial ∧ ¬ human and ‘boy’, ⇒ mammal ∧ terrestrial ∧ human.
Ontological Semantics, Another way of decomposing the meaning of a word is by studying its relationship to other words, Take the classic example, of the word ‘bank’, ‘Bank’ itself. means a collection or storage. However, what that collection is about – ‘water’, ‘blood’ or ‘money’—dictates the exact definition that word will semantically adopt. This process of defining the existence/usage of a term with respect to a sentence is called ontology, WordIer (Miller 1995) is a famous lexical and ontological resource in English It contains various kinds of relations.
2.3.4 that exist between English words, For example, the word small, might be synonymous with little while it conveys the opposite meaning to large The notion of a mouse implies, that it is a type of animal. Capturing these relations.
2.3.4 in the text, is crucial to understanding world knowledge.
Distributional Semantics, So far in our discussion of semantics we have assumed the computational methods to carry the same level. of contextualization as humans While machines lack subconscious contextualization they can approximate the same by analyzing large corpora. of text, and deriving a sense of words, based on their distributional properties egg co-occurrence, frequency This maps to the law of association that words, with similar distributions might have similar meanings. For instance, the meaning of the word mouse may be complex for the machine to grasp, yet it can be inferred from the contexts, it appears in id sentences where it cookers with words, like ‘rodent’, animal. ‘food’, etc.) Distributional Semantics, forms the core of the modern-day ALP Introduction to Language Modelling
According to Herbert Clark, whenever two words, occur together or in close proximity, an associative link is formed between them. in our mind over time, and the more frequently they appear together the stronger the association Clark, Building. up word association and logic of distributional semantics we can describe a Language Model KM as a model that learns the probability distribution over the words, in the corpus; This probability is learned based on the frequency co-occurrence, of words, in a large training corpus; Once trained/learned, the KM attempts to predict the next token in a sequence of tokens, For a sequence of ‘m’, tokens, x1, x1, . . ., cm the KM predicts the ‘m’, + token cm based on the language learned from its training corpus; of words, and phrases The output space. id the set of all possible. words, that can be the ‘m’, + eh token in a sequence is the whole vocabulary/lexicon learned over the language If the KM is learned over N: unique tokens, then in the worst case, each of N: tokens, has an equal and independent probability of N: for being the ‘m’, + eh token.
However, from our semantic and syntactic parsing we know that for a given sentence not all words, have an equal probability of occurrence Instead the words, that can appear next are conditioned on the words, that are present so far in the sentence It forms the basis of language modeling in ALP In layman terms a language model predicts the probability of the ‘m’, + eh token given a sequence of ‘m’, tokens, seen before Going back to our example, sentence if you are asked to predict the next word in the sequence of ‘Hello Sam. How are of all the words, we know in English id our vocabulary the most likely next word should be you This likelihood is the probability spread over the whole vocabulary of which you has the highest probability score. We will introduce the formal concepts of conditional probability and language modeling in detail in Chapter 4.
Bag-of-Word Based Representation. Forgoing the notion of conditional probability one can still obtain a crude form of language modeling that depends solely on the constituted tokens, present in the sentence Let us. consider the task of sentiment analysis A simple method for determining whether a sentence expresses positive, sentiment ‘would’ be to count the favorable and negatively connotated lexical terms that occur in the sentence The process is solely based on the occurrence of individual words, and not where and how they appear in the sentence id the notion of semantics or syntax is overlooked. Such setups are called the bag-of-word approach. where we know the words, in the bag but not the order in which they are placed in the bag.
Example 2.2. Let us. understand the bag-of-words modeling via a simple example, of sentiment classification.
Consider three sentences that represent three samples of sentiment analysis S1: The movie is bad.’, S1: The movie is ‘good’ S1: I, liked the movie’.
After reprocessing lowercasing, (punctuation). removal, lemmatisation. liked → like and tokenisation we end, up with a engram vocabulary set the movie is bad.’, ‘good’ I, like Based on the unique vocabulary the sentences can then be represented as vectors of length 7, indicating whether the it index vocabulary term is present in the sentence or norTh's S1: = [yes, [yes, [yes, [yes, no no no and mapped numerically as [1, 1, 1, 1, 0, 0, 0] where 1 means seethe token is present and 0 means note token is not present in the given sentence In a similar way S1: and S1: become [1, 1, 1, 0, 1, 0, 0] and [1, 1, 0, 0, 0, 1, 1], respectively.
Further, each sentence has a sentiment label, associated with it where –1 means negative sentiment 0 means neutral. and 1 means positive, Our example, sentences have a sentiment score. of S1: –1, Sb and S1: 1, respectively. From the crude analysis of the sentence vectors we (see that tokens, the and movie occur in all three sentences and do not lead to any differentiation for the sentiment classification id we cannot tell by looking at only these two terms if the movie is ‘good’ or bad.’, Meanwhile, the presence of bad.’, in S1: and its subsequent absence in S1: and S1: is an indicator of associating the presence of bad.’, with the label, –1. Language models build on bag-of-word representation and try to learn such heuristics between tokens, and labels based on the frequency of occurrence of the tokens, in different class labels.
The notion of building representations from term frequency is detailed in Chapter Part II, Neural Networks
So far the algorithms we have discussed for parsing and understanding language are based on simple heuristics and probabilities. To develop a more advanced and nuanced understanding of language we must work with neural networks This part of the book will help readers establish a basic understanding of computational neural networks The theoretical foundation for these networks can be traced back to the independent works of Alexander Cain in 1873 and William James in 1890. Both hypothesized that human thoughts and decisions emerge from interactions among billions of neurons in the human brain. This biological network of nerve cells is responsible for all human reasoning and decision-making. Warren McCulloch, a neuroscientist, and Walter Pitts, a logician, laid out a theoretical model for a biological nerve cell in 1943. They called it a perception In 1957, Frank Rosenblatt, a psychologist, provided an early hardware implementation of a perception Rosenblatt, took a linear combination of different input.
2.2 variables and gave a response. of 1 or 0, depending on whether the linear combination of input.
2.2 variables was, positive, or negative The version of the perception we use today was, introduced by Minsky and Paper in 1969. They introduced the concept of an activation function an essential component of all artificial neural networks used today The Perceptron
The architecture, of computational neural networks is inspired by the nervous system in humans where a network of neurons is responsible for processing relaying, storing, and recalling information A biological neuron receives signals from other neurons and chooses to transfer the processed signals to neighboring neurons depending on the outcome of processing We can replicate the same in software via the perception which is the most straightforward software implementation of a biological neural cell Definition
Given a Dimensional input.
2.2 vector. x1, = x1, x1, …, xV the perception computes a linear combination ax + ax + …+ wan adds a term β and decides to output among the values {–1,0,1} depending on the computation. Formally, a perception can be represented by Equation (2.3), where ‘w’, = ‘w’, ‘w’, …, aN is called the weight vector. β is called the bias = sen wAx + where sen is the signal function defined asCots that the sen is a step function We will slightly modify this function in the next section, to model some elementary boolean functions. Implementing AND, OR, and XOR Logic
Given that any computational task can be decomposed into a combination of Boolean operations, exploring the scope of modeling such functions. using the perception is highly motivated We will attempt to model some elementary boolean functions. using the perception defined in the previous section, with a slightly modified definition of sen function In particular we will model AND, OR, and XOR Boolean functions. their function definitions are shown in Tables 2.4, 2.5 and 2.6, respectively. These are binary functions. as they take two input.
2.2 variables denoted by x1, and x1, and the output denoted by y, within one of the possible. values {0, 1}.
x1
x2
x1 AND, sTable 2.4: The AND, Function.
x1
x2
x1 OR, sTable 2.5: The OR, Function.
x1
x2
x1 XOR sTable 2.6: The XOR Function.
The AND, function or gate) implements logical conjunction. It takes two Boolean inputs either 1 or 0) and produces an output according to the Truth Table 2.4. We can model the AND, function using a perception where we have to assign such values to ‘w’, ‘w’, and β so that Equation (2.4) below satisfies Table 2.4 with y, = x1, AND, by = sgn′(w1x1 + ax + where x1, x1, ∈ {0,1} and sen is defined reInsert Figure]
Figure 2.6: D, plots showing different boolean logic functions. and the corresponding line (dotted) separating the input.
2.2 coordinates with different output values for AND, (Left), OR, Center and XOR (Right). Note that no separating line exists for the XOR function.
The D, plot for the AND, function in Figure 2.6 (Left), shows the line x1, = x1, + 1.5 linearly separating the input.
2.2 coordinate points associated with 1 and 0 output values If we let ‘w’, = 1, ‘w’, = 1, and β = – 1.5, then the perception model y, = sax + x1, – 1.5) emulates the AND, gate) as y, attains 1 if and only if both the inputs x1, and x1, assume value 1. We can verify this from Table 1}.
x1
x2
x1 + x1, – y, = sax + x1, – x1, AND, sTable 2.7: The perception model y, = sgn′(w1x1 + ax + β) with ‘w’, = 1, ‘w’, = 1 and β = –1.5 correctly models the Boolean AND, function.
The OR, gate) or function implements logical distinction It receives two Boolean inputs either 1 or 0) and produces an output according to the Truth Table 2.5. Following the same perception model defined in Equation (2.4), what values should the weight and bias be assigned? The D, plot for the OR, function in Figure 2.6 Center shows the line x1, = x1, + 0.5 linearly separating the input.
2.2 coordinate points Let ‘w’, = 1, ‘w’, = 1, and β = –0.5. Then, the perception model y, = sax + x1, – 0.5) emulates the OR, gate) as y, attains 0 if only if both the inputs x1, and x1, assume value What is happening with XOR If we observe Figure 2.6, we (see that the AND, and OR, functions. possess a linear boundary separating the points labeled with output values For XOR no such boundary exists If we try to model the XOR function using the perception definition in Equation (2.4), we will fail to model it.
2.9 Multilayer Perceptron
To be able. to model more complex functions. we need to generalist the perception architecture, Let us. define a more general neuron-like processing unit where we replace sign(·) with a generic function ϕ(·) termed as the activation function or transfer function A neural network is realized as a combination of such neuron-like processing units as formulated in Equation where oi is the it component of the weight vector. xi is the it component of the input.
2.2 vector. and β is the bias term Note that the output ‘z’, is also termed as the hidden unit More specifically, in this chapter, we will learn about feed-forward neural networks where these units are combined in a tree-like fashion without any cycles.
The most straightforward feed-forward neural network is the Multiplayer Perception ALP as shown in Figure 2.7. Here, the neuron-like units are arranged in a set of layers, with each layer having some number of these identical units The first layer is called the input.
2.2 layer and the units in this layer receive the input.
2.2 features, The last layer is called the output layer and the number of units in this layer can vary depending on the output required from the feed-forward model All the layers, in between are called the hidden layers, The number of layers, is known as the depth, and the number of units in a layer is known as the width of that layer As you might have guessed, deep learning refers to training neural networks with many hidden layers.
[Insert Figure]
Figure 2.7: Architecture of a Multiplayer Perceptron.
[Insert Figure]
Figure 2.8: Implementing XOR Boolean function using an ALP with a single hidden layer and sen as the activation function as defined in Section 2.8.2.
Can an ALP model XOR function 2.8.2.
Can combining multiple perceptions help in modeling the XOR function Figure 2.8 shows the required ALP architecture, The XOR function returns 1 when exactly one of the inputs is 1. We can use hidden units to capture this information Let ‘h’, sax + x1, – 0.5)) detect if at least one of the input.
2.2 features, is 1 and let ‘h’, sax + x1, – 1.5)) detect if both the input.
2.2 features, are 1. The output y, will then be one if and only if ‘h’, = 1 AND, ‘h’, = 0. From Table 2.8, we can (see the values the hidden and output units attain at various values of input.
2.2 features.
Input
Units
Hidden Units
Output
x1 XOR xxx = sax + x1, – ‘h’, = sax + x1, – y, = shh – ‘h’, – sen · 1 + 0 · 1 – 0.5) = sen · 1 + 1 · 1 – 0.5) = sen · 1 + 0 · 1 – 0.5) = sen · 1 + 1 · 1 – 0.5) = sen · 1 + 0 · 1 – 1.5) = sen · 1 + 1 · 1 – 1.5) = sen · 1 + 0 · 1 – 1.5) = sen · 1 + 1 · 1 – 1.5) = sen · 1 + 0 · – 1 – 0.5) = sen · 1 + 0 · – 1 – 0.5) = sen · 1 + 0 · – 1 – 0.5) = sen · 1 + 1 · – 1 – 0.5) = Table 2.8: Modeling the boolean XOR function using the Multiplayer Perception in Figure Neural Networks
We will now formally define a neural network Assume a neural network that takes an Dimensional input.
2.2 vector. x1, x1, ..., xV and outputs a Dimensional vector. y, y, ..., bK The neural network contains a single hidden layer depth, = 1) with M neurons in the layer width = M realized as a linear combination via Equation (2.6), where j) = 1, 2, . .., Mae refer to the parametersas weights and as bias following the terminology we laid out in the definition of the perception in Equation (2.3). The quantities are known as activation Each of them. is transformed using an activation function ‘h’, as . In the context, of neural networks is termed as the hidden units Following Equation (2.6), the hidden units are again linearly combined with suitable weights and biases to give output activation in Equation (2.7) where ‘k’, = 1, 2, ..., K, and K, is the total number of output variables Finally, the output activation are transformed using activation function ‘h’, which may be the same as ‘h’, to produce the output values bk as . We can combine Equations (2.6) and (2.7) to write the overall neural network function as follows:
(2.8)
Succinctly, as Equation (2.8) describes a neural network is simply a function that maps a set of input.
2.2 variables x1, x1, xV to a set of output variables y, y, bK using a series of controllable parameters: weights and biases The forward processing of input.
2.2 variables in order to generate the set of output variables is termed the forward propagation of the neural network Sigmund The sigmoid/logistic activation σ(·) is defined Ashe Sigmund activation function Equation 2.9) is generally used when we have to model probability as the output since the range of this non-linear function is between 0 and 1. The derivative of the sigmoid activation function ‘f’, (·) can be written in terms of the function of itself. as x1, = x1, stanch The tang activation function tang is defined Ashe output of the tang activation function Equation 2.10) is zero-centred within the range of –1 to 1. Hence, we can easily map output values as strongly negative neutral. or strongly positive, The derivative can be written as tangs = 1 – tanh2(x).
3. Softmax: So far we have looked at the activation function which models only a single output at a time, If we want to classify a data point into one of many categories or classes, then we employ the softmax activation function It outputs a well-defined probability distribution over all the output classes, and is generally used in the output layer of the neural network The Softmax activation is formally defined example 2.3. Consider a list with frequency count as [1, 5, 2]. Converting this list into probabilities. represented by softmax will require the denominator to be D, = ‘e’, + ‘e’, + ‘e’, and each frequency in the list can then be represented as [e1/D, [e1/D, [e1/D, or [0.02, 0.94, 0.04] whose sum approaches unity.
4. ReLU: ReL stands for Rectified Linear Unit and is defined as:
ReLU(x) = max the ReL activation function Equation 2.12) is far more computationally efficient than the sigmoid or tang activation functions. This is because only a selective set of neurons are activated when ReL is employed.
5. GELU: GEL stands for Gaussian Linear Unit and is defined by Equation (2.13), where x1, is the cumulative standard normal distribution.
GELU(x) = x1, · the GEL activation function weights the incoming signal by their percentile rather than their sign(·) Consequently, GEL can be thought of as a smoother ReLU.
6. GLU: BLU stands for Gated Linear Unit The linear gating is parameterized via a signori activation Equation 2.9) on weight ‘w’, and bias ‘b’, as defined in Equation GLUm = x1, ⊙ ax + the operation ⊙ represents the component-wise multiplication, and this allows the BLU to control the flow of information from the incoming vector. x1, by learning which parts. to emphasis or de-emphasise.
7. Swish: The Swish activation function also employs sigmoid activation to induce smoothness and differentiability. The Swish activation function Swish is defined as:
Swish(x) = x1, · where β is a learnable parameter.
8. SwiGLU: SwiLl stands for Swish-Gated Linear Unit Activation Function. As the name suggests it combines Swish and BLU via Equation (2.16) and allows for better optimization of weights.
SwiGLU(x) = x1, ⊙ Swished + ‘b’, Training Neural Networks
Now that we have established. neural networks to be parametric non-linear mapping functions. the question: remains how do we assign values to the network parameters: id weights and biases We will elaborate on this in this section.
(2.17)
After each update, the gradient is re-evaluated for the new weight vector. and the process is repeated. The error function is defined with respect to a training set so each step/iteration requires that the entire training set be processed in order to evaluate Ea This algorithm of training captured by Equation (2.17) is known as the gradient descent. Backpropagation
Error functions. usually comprise a summation of the error over each data point of the training set . This reduces our problem to evaluating the gradient with respect to only one training sample here nth training sample Eh insert Figure]
Figure 2.9: A basic neural network architecture, with linear activation function and no hidden layers.
Referring to the neural network shown in Figure 2.9, the kph output unit bk where ‘k’, ∈ {1, 2, ..., K, is a linear combination of the input.
2.2 variables xi ‘i’, ∈ {1, 2, ..., N: such that . As the process of gradient descent. involves obtaining the derivative of the error function art to the weights we prefer an error function that is continuous and differentiable, and the squared error is one of the most straightforward functions. that fit this criterion. The squared error term for the nth training sample can be written as follows:
(2.18)
The gradient of this error term with respect to weight win is given by Equation (2.19). Note in the above example, win connects the activation node. associated with the it input.
2.2 feature to the nth output value In general for a hidden layer win connects the it unit of the previous layer to the nth unit of the next layer The weight gradients in Equation 19 can then be used to update, the weight win according to Equation This process of calculating error gradients by using the chain rule starting from the output layer to the hidden layers, is termed as backpropagation.
Example 2.4. Let us. understand more about backpropagation. using a more general example, involving a neural network with a single layer of hidden units and the tang activation function which we saw in Section 2.9.2. It receives Dimensional input.
2.2 x1, = x1, x1, . . . , xV and outputs a Dimensional vector. y, = y, y, ..., bK The hidden layer has M hidden units Refer to Figure 2.10 for the neural network described above in this figure, the output units have a linear activation function bx = x1, and the units in the hidden layer have tang activation function The final output bk can be expressed as , with and . At the input.
2.2 level. 2014).
[Insert Figure]
Figure 2.10: A neural network architecture, with a single hidden layer and a non-linear activation functionEd consider the standard sum of error squares as the error function For nth training sample let y, = bk denote the predicted output and ‘t’, = pk denote the target for The error term for the nth training sample can thus be represented as follows:
(2.20)
We can calculate the required gradients for backpropagation. via Equations (2.21) and Sand Batching
We have already described in the gradient descent. algorithm in Equation (2.17) that prior to weight update, a full pass through the training datasets is required In the case, of a large number of training samples it becomes computationally expensive and slow to iterate over all the data samples at once. In order to improve the training process we use two variants of gradient descentS Stochastic Gradient Descent
This variant of gradient descent. allows for the sedation of the model parameters: after processing a single training example.
(2.24)
This variant allows for faster convergence towards the minima and is less memory-intensive (loads only a single sample to memory at a time, than vanilla gradient descent. However, by optimizing after each sample the model is more likely to overfit.
In the Stochastic Gradient Descent STD algorithm stochastic or randomness comes into play. when we randomly select a data point to be optimized While in each epoch, all the data points are processed the order in which they can be processed can be randomly shuffled.
Mini-Batch Gradient Descent
In the case, of the vanilla gradient descent. for N: number of samples and ‘T’, epochs, the weight optimization operation happens only ‘T’, times. In STD the optimization operation happens N: × ‘T’, times. Between optimizing one sample at a time, vs optimizing all samples aggregated, we can update, the gradient over a group of samples instead Let N: samples be grouped into a set of ‘n’, smaller samples The optimization step is performed ‘n’, × ‘T’, times. such that ‘T’, < ‘n’, × ‘T’, < N: × ‘T’, This optimization technique is called mini-batching, and each of the ‘n’, sets is called a batch, denoted as Example 2.5. Consider the neural network in Figure 2.11. The network takes two input.
2.2 variables x1, and x1, outputs two continuous variables y, and y, and utilities the Sigmund activation function at each hidden unit At current training checkpoint, the weights have following values . The bias terms ‘b’, = 0.25 and ‘b’, = 0.35.
Given a new training input.
2.2 vector. x1, = x1, x1, = (0.1, 0.5) and the expected output ‘t’, = ‘t’, ‘t’, = (0.05, 0.95), let us. calculate the update, for using stochastic gradient descent. and η = We will first forward propagate through the neural network to store values of hidden units and predicted outputs.
Forward Propagation:
[Insert Figure]
Figure 2.11: The Neural Network Architecture for Example We will now calculate the error contribution due to this new training input.
2.2 vector. Hyperparameters
As explained before the training of a neural network involves processing all the samples in the training datasets for which the model is optimized Once trained id no more weights are updated it is imperative to determine how well the model will predict on unseen samples The datasets on which we evaluate the generalisability of a trained neural network is called the test datasets Note we assume that both training and testing samples are drawn from the same underlying distributionS neural network model is said to underfeed if it fails to perform well even in the training stage. It can be a result of the smaller number of training samples from which to learn any meaningful patterns or the smaller complexity of the neural network that prevents it from learning more complex patterns within the training datasets or both On the contrary, a neural network is said to overfeed if it performs well on the training datasets but fails to perform on the test set In such cases, a neural network learns the noisy patterns in the training set which leads to a lack of generalisability.
Thus, by controlling how complex the network is and configuring the learning rate η, we can in turn, impact the learning process Such configurable variables explicitly. declared before training whose value controls the learning process are termed hyperparameters.
<H4> Breadth and Depth
Based on our understanding of overfitting and underfitting, it appears that the models complexity plays a vital role in the learning process ‘But, how do we define the complexity of a neural network In terms of the number of weight multiplication, operations, that form the basic building block of a neural network we can control the complexity of the network by capping the number of activation units Recall the concept of depth, and breadth of an ALP in Figure 2.7. By increasing the depth, of the network we allow the system to model more complex functions. Meanwhile, by increasing the breadth of the network we can accommodate more feature vectors Both will enable us. to reduce underfitting, Note that while theoretically, one can have infinite depth, and breadth such a system will overfit.
<H4> Number of Epochs
The ideal number of training iterations/steps is such that any further training provides little to no boost in test accuracy The number of iterations is also known by the term number of epochs, where each epoch, is complete when all the training samples have been processed.
<H4>Learning RatTle learning rate η determines the magnitude of steps. taken in the direction of decreasing gradient Equation 2.17). A large learning rate implies, ‘taking’ larger strides, which may lead to scenarios where we keep hovering around the local minima without reaching it In contrast, with a smaller learning rate it takes too long to reach the optima. There are various strategies that one can use to manage the learning rate during the training of a neural network Fixed Learning Rate: In this training strategy, the learning rate remains constant throughout the training process 2. Time-Based Decay: In this training strategy, the learning rate decreases proportionally to training steps. It is based on the idea that initially the model will begin by predicting randomly and have a higher error rate However, as the training progressed, the error ‘would’ have reduced , where decay is a factor by which the learning rate decreases and epoch, is the training iteration. ‘t’, Regularisation
Regularisation is another. set of techniques that can help avoid overfitting during training.
Early Stopping: One of the most straightforward techniques to prevent overfitting is to limit the number of updates made to the weight parameters: Heuristically, if we can avoid the training loss from becoming arbitrarily low, the model will be less likely to overfilL and L2 Regularization By penalizing larger weights while training we can further reduce overfitting Let us. first look at the Le norm of a vector. x1, in an dimensional space. defined by . When p = 1, we call this the L2 norm or Manhattan distance given by and when p = 2, we refer to it as the L2 norm given by .
Employing the penalty term we can minimize the error term Ea via Equation (2.26) with α the regularization constantLy replacing p with 1 or 2, we obtain the L2 or L2 regularization respectively. L2 regularization allows for more sparse weight parameters: Unlike L2 regularization that forces weights to zero, L2 regularization shrinks weights while ensuring that important components of the weight vector. are larger than the others.
Dropout: As the name suggests we randomly drop or freeze a fraction (dropout probability of neurons from being updated Suppose we are using minibars gradient descent. using a (dropout regularization ‘would’ amount to training different weight parameters: for various subsets of training data to avoid overfitting the entire training datasets During test time, no neurons are dropped.
2.11 Vanishing and Exploding Gradients
When obtaining the derivative of the loss with respect to weights the private value may be extremely small, or large leading to the problem of vanishing or exploding gradients.
Vanishing Gradients. This refers to the situation when the gradient information cannot be transferred from the output layers, to the hidden layers, due to the gradients assuming very small, values Following our previous notation, let L2 denote the index of the output layer Then, we calculate the gradient of error term Ea with respect to weights in different layers, of the networks For the it hidden layer let us. denote the weight parameters: as oi the hidden units as hi and activation as AI such that hi = haj where ‘h’, is the activation function Note that each hidden layer ‘would’ have multiple hidden units but we do not label, such hidden units to avoid complications.
As we note from Equation (2.27), the further the hidden layer ‘i’, is from the output layer (deeper the neural network the more terms of the form incorporating the partial derivative of the hidden unit with respect to the activation appareNt so happens that these derivatives assume very low, values for activation functions. like sigmoid and tang ReL activation is usually employed when there is a risk of a vanishing gradient problem.
Exploding Gradients. On the opposite spectrum is the problem where large error gradients accumulate and result in huge updates to neural network model weights during training These may occur due to lousy initialization of weights or some combinations, of activation functions. Evaluation Metrics
Once we have optimally trained our neural network we need to be able. to report how well the model is performing Additionally, given that for a given set of input.
2.2 and target values multiple optimal weights can be obtained How do we determine which set of weights are the ‘best’ for an unseen datasets To perform this assessment, we utilize evaluation metrics.
Let us. go’ back to the task of sentiment analysis Suppose we have ten sentences that are labeled as either positive, (1) or negative (–1). Out of these seven samples are labeled as positive, Let us. assume an arbitrary target label, list for the ten samples as y, = [1, 1, –1, 1, –1, –1, 1, 1, 1, 1], with the it element of the list providing a sentiment label, for the it sentence.
Case How many times. did we correctly predict the positive, sentiment?
Case How many times. did we incorrectly predict positive, sentiments as negative?
Case How many times. did we incorrectly predict negative sentiments as positive?
Case How many times. did we correctly predict the negative sentiment?
True Positive/Negative. Case 1 of the confusion matrix can also be termed as true positive, GP as we are truly/correctly predicting the positive, class as positive, Consequently, case, 4 is termed as true negative (TN) as we truthfully predict the negative class as negative.
False Negative. Case 2 can be understood as the number of times. we erroneously/falsely produce a negative output sentiment in our case, when the actual output is positive, id false negative False Positive. Reverse of FM is when we falsely predict the output to be positive, while it should have been negative leading to the case, of false positive, Example 2.6. Let us. map true positives, true negatives, false positives, and false negatives, when y, = [1, 1, –1, 1, –1, –1, 1, 1, 1, 1] and ŷ = [1, –1, 1, 1, –1, 1, 1, 1, 1, –1]. Further, based on these counts we can produce a confusion matrix.
In Table 2.9, we enlist the type of correct/incorrect information captured by the it index We can (see that GP occurs when bi = ‘i’, = 1 and (TN) at bi = ‘i’, = –1. Meanwhile, at indices 2 and 10, we observe the case, of bi = 1 but ‘i’, = –1, causing false negatives, Finally, at indices 3 and 6, we note ‘i’, = –1 but ‘i’, = 1, leading to false positives.
Now, mapping the type count in Table 2.9, we can construct the confusion matrix for the four cases, as accounted in Table 2.10.
Precision. Looking only at the predictions that are marked as positive, precision measures the number of times. the predictions were actually correct as actualized by Equation (2.28).
(2.28)
Index
1
2
3
4
5
6
7
8
9
10
Expected predicted ŷ
1
-1
1
1
-1
1
1
1
1
-1
Type
TP
FN
FP
TP
TN
FP
TP
TP
TP
FN
Table 2.9: Mapping True Positives GP True Negatives (TN) False Positives DP and False Negatives FM for Expected Labels y, = [1, 1, –1, 1, –1, –1, 1, 1, 1, 1] and Predicted Labels ŷ = [1, –1, 1, 1, –1, 1, 1, 1, 1, –1].
Actual
Predicted
Positive
Negative
Positive
5 GP Negative DP EaTable 2.10: Confusion matrix for sentiment classification of positive, (1) and negative (-1) sentiments for ten sentences We construct this from expected labels y, = [1, 1, –1, 1, –1, –1, 1, 1, 1, 1] and predicted labels ŷ = [1, –1, 1, 1, –1, 1, 1, 1, 1, –1]. The tabulations follow? from mapping in Table Recall On the other hand. looking at the actual/expected positive, samples recall measures the number of times. we correctly predicted the positive, class The confusion matrix can be mapped using Equation 2.10.
Precision. vs Recall We note from Equations (2.28) and (2.29) that the main difference in precision and recall is dictated by the type of erroneous outputs that are accounted for In the case, of precision we place higher importance on FPs. Consider the case, of spam detection. If the emails keep getting falsely classified as safe/positive, then the user will be inundated with spam instead of useful. information Meanwhile, in the case, of a recall we place higher importance on FMs Consider the case, of medical testing where a positive, test means a disease is detected. Failing to detect the disease FM when it should have been positive, can cost human life. In any given experimental setup. precision and recall will be a tug-of-war, as reducing FM can impact DP and vice-versa, and which metric is prioritized depends on the task at handS Score. For most use cases, we rather prefer to look at a single metric that considers both precision and recall Here, the F1 score. comes into play. It is simply a harmonic mean of precision and recall as follows:
Another advantage of the F1 score. over other metrics is its ability to account for class imbalance, therefore providing a more holistic measure of model performance Summary
In this chapter, we explored some of the fundamental concepts of ALP and neural networks necessary for understanding the more advanced topics covered later in the book We began by discussing the motivation behind processing information conveyed. through natural language focusing on how a word is structured using morphological knowledge We then reviewed the essential steps. of the ALP pipeline. and examined various reprocessing techniques such as stemming lemmatisation. and tokenisation Additionally, we explored the syntax and semantics of language before introducing core ideas related to language models and word/sentence representation techniques.
To motivate the use of dimensional feature vectors for sentiment analysis we introduced the concept of neural networks 0.35.
Given that neural networks are the foundation of modern ALP this chapter, provided an overview of the fundamental aspects of neural networks We discussed perceptions and their limitations, which led to the development of multiplayer perceptions and the concept of deep neural networks The chapter, also covered training neural networks via backpropagation. the basics of activation functions. and the role of various hyperparameters that can impact the training process Furthermore, we outlined scenarios where a model might encounter vanishing or exploding gradient problems and how these issues can be mitigated. The chapter, concluded with an introduction to evaluation metrics commonly used in classification taskIng the following chapters. we will build upon the concepts of word associations, neural networks and grams to develop more sophisticated representations and language models that go’ beyond the bag-of-words approach.
Additional Resources
Important Articles Survey of Surveys ALP & ML): https://github.com/NiuTrans/ABigSurvey.
Awesome ALP https://github.com/keon/awesome-nlp.
Introduction to Linguistics Akmajian er all Milks Sabrina J., er all Between Words and Characters: A Brief History of Open-Vocabulary Modeling and Tokenization in ALP lXiv reprint lXiv (2021).
Min, Conan er all “Recent advances in natural language processing via large retrained language models A survey.” ABM Computing Surveys 56.2 (2023): 1-40.
Otter, Daniel W., er all A survey.” of the usages of deep learning for natural language processing IEEE transactions on neural networks and learning systems 32.2 (2020): 604-624.
Pattern Recognition and Machine Learning Bishop (2006)
Visual Summary
Dependency Parsing Named Entity Recognition Tokenization and Token Similarityhttps://huggingface.co/spaces/spacy/pipeline-visualizer#en\_core\_web\_lg
Deep Neural Network Architecture https://playground.tensorflow.org/
Optimization with Gradient Descenthttps://uclaacm.github.io/gradient-descent-visualiser/#playground
Exercises
True/False Questions
Lemmatisation is more computationally expensive than stemming (True/False)
The sigmoid activation function outputs a value between -1 and 1. (True/False)
SentencePiece does not require the input.
2.2 sequence to be pre-tokenised. (True/False)
Multiplying the output of a linear unit with a scalar can introduce non-linearity. (True/False)
Dependency parsing focuses on identifying relationships between words, based on the order in which they appear in a sentence (True/False)
Multiple Choice Questions
1. In dependency parsing the \_\_\_\_\_\_\_\_\_\_ is the main (verb) of the sentences Prime ROTC Lemmas SeeThe study of the internal structure of words, is called a Etymology Sociolinguistics (c) Morphology Phonology
3. In gradient descent. what is updated out of the followings Parameters Inputs Architectures Activations
4. Dependency parsing helps. in understanding the \_\_\_\_\_\_\_\_\_\_ structure of a sentences Syntactic Semantic Pragmatics Morphological
5. Which of the following introduces non-linearity. into a neural models Weight Sharing Gradient Descent Convolutions GEL Activation
Short Questions
How does stemming reduce the dimensionality of textual data?
What is the difference between stemming and lemmatisation?
Compare the ReL and sigmoid activation functions. ‘Which one of them. is used in ‘i’, the hidden layers, and (ii) the output layer Why?
Consider the following sentences Try to trace an ALP pipeline. that consists of tokenisation POSS tagging, lemmatisation. and dependency parsing on each sentenceD am eating pizza with cheese and cornY mother cooked my favorite dish for me on my birthdayS am at the airport, and my flight departs in an gourDe loves to bake cookies for his friends and family.
Long Questions
Compare and contrast, various tokenisation strategies discussed in this chapter.
Explain how text, reprocessing techniques impact the performance of ALP models.
Calculate the output of a three-input neuron where the weights wow ‘b’, are [0.3, –0.1, 0.2, 0.5]. The input.
2.2 to this network is [0.3, 0.2, 0.6]. Assume the sigmoid activation function.
What is WordIer Explain the structure and applications of WordNet.
Why do we need sword tokenisation Give an example, where word tokenisation fails.
Describe the typical stages involved in a natural language processing pipeline. Explain the significance of each stage. by ‘taking’ some ALP tasks. as an example.
Consider a simple neural network with one layer and sigmoid activation where ŷ = wAx + ‘b’, and ‘z’, = 1/(1 + expo Compute the gradients of the loss function with respect to the parameters: and derive the weight update, rule for gradient descent.
Bibliography
Akmajian, A Demurs R. A Farmer, A K, and Garnish R. M (2001). Linguistics 2
An Introduction to Language and Communication. MIT PressURe https://doi.org/10.7551/mitpress/4252.001.0001
Bishop, C. M (2006). 604-624.
Pattern Recognition and Machine Learning 4.
Information Science, and Statistics). Berlin, Heidelberg: Springer-Verlag.
Cho, K, van Merriënboer, B., Gulcehre, C. Bahdanau, D, Cougar's F1 Schwinn H., and Benito Y. (2014). Learning phrase representations using RN encoder–decoder for statistical machine translation In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing EMILE (pp. 1724–1734).URL https://aclanthology.org/D14-1179
Clark, H., H., (1970). Word Associations and Linguistic Theory. New Horizons in Linguistics 1, 271–286.
Eisenstein, J., (2019). Introduction to Natural Language Processing MIT Press.
Elman, J., L2 (1990). Finding Structure in Time. Cognitive Science, 14(2), 1724–1734).URL https://www.sciencedirect.com/science/article/pii/036402139090002E
Gage, P (1994). A New Algorithm for .
Data Compression. The C. Users Journal, 12(2), Gees F1 A Schmidhuber, J., A and Cumming F1 A (2000). Learning to forget: Continual prediction with lam Neural Compute 12(10), 2451–2471.