Natural Processing Language

UNIT 2

By Shivani Deopa

Semantic and Word Embedding Semantics

Words and Vector

- Words that occur in similar contexts tend to have similar meanings.
- The meaning of a word is related to the distribution of words around it.
- Imagine you had never seen the word tesguino.
- Let’s consider the following 4 sentences:

A bottle of tesguino is on the table.

Everybody likes tesguino.

Tesguino makes you drunk.

We make tesguino out of corn

- From these sentences, you can figure out tesguino means a fermented alcoholic drink like beer, made from corn.
- We can capture this same intuition automatically by just counting words in the context of ino; we’ll tend to see words like bottle and drink.
- These words and other similar context words also occur around the word beer or liquid or tequila.
- This can help us discover the similarity between these words and tesguino.
- The meaning of a word is computed from the distribution of words around it, such method is called distributional method.
- These words are generally represented as a vector or array of numbers related in some way to counts.

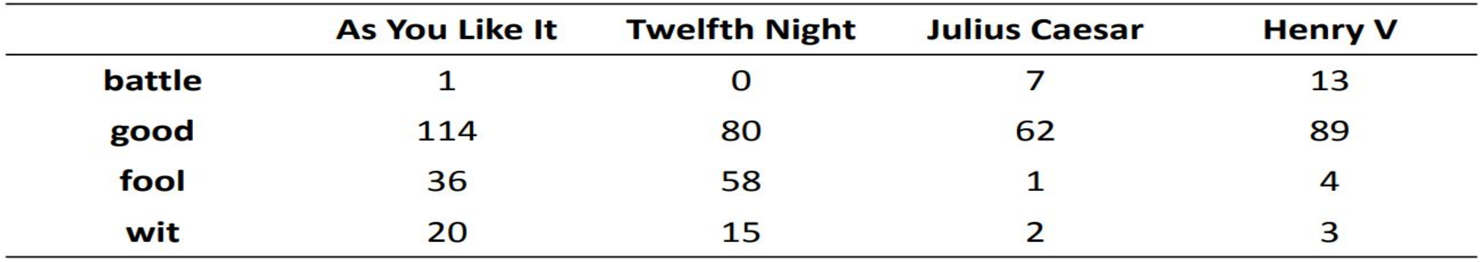
Vector semantics:

– distributionalist intuition

– vector intuition

- A simple method in which the meaning of a word is simply defined by how often it occurs near other words, this method results in very long vectors that are sparse.
- We will expand on this simple idea by introducing a way of constructing short, dense vectors that have useful semantic properties.
- The idea of vector semantics is thus to represent a word as a point in some multi-dimensional semantic space.
- The shared intuition of vector space models of semantics is to model a word by embedding it into a vector space.
- The representation of a word as a vector is often called an **embedding.**
- Vector or distributional models of meaning are based on co-occurrence matrix – a way of representing how often words co-occur.
- In a term-document matrix, each row represents a word in vocabulary and each column represents a document

- The following table shows a small selection from a term-document matrix.



	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- The matrix shows the occurrence of fours words in four plays by Shakespeare.
- Each cell in the matrix represents the number of times a particular word occurs in a particular document
- Each position indicates a meaningful dimension on which the documents can vary.
- The first dimension for both these vectors corresponds to the number of times the word battle occurs, and we can compare each dimension.

- We can think of the vector for a document as identifying a point in |V| -dimensional space.
- The documents are points in 4-dimensional space.
- Generally, the term-document matrix X has |V| rows and D columns.

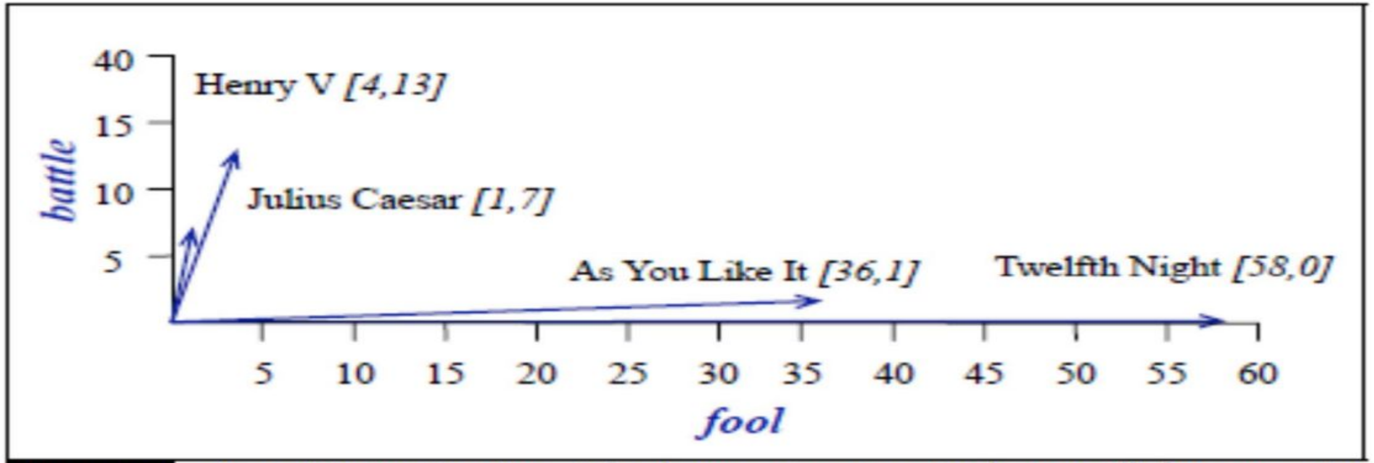
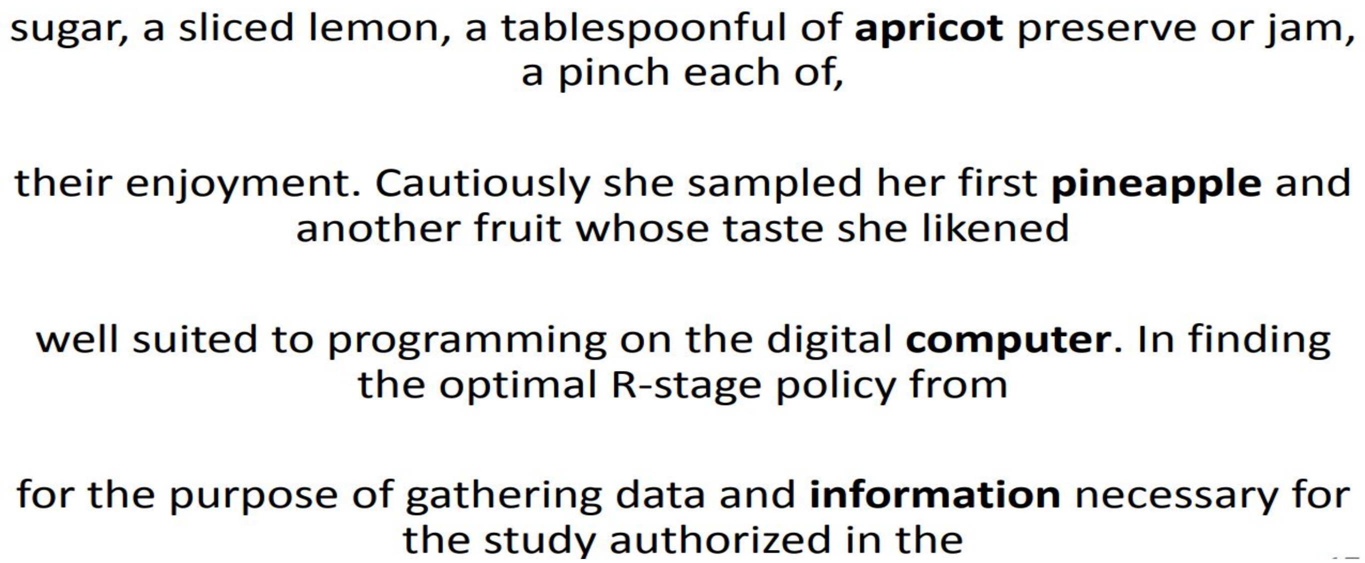


Figure 6.4 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

- We’ve seen that documents can be represented as vectors in vector space.
- However, vector semantics can also be used to represent the meaning of words, by associating each word with a vector.
- The word matrix is now a row vector rather than a column vector, hence the dimensions of the vector are different.
- The four dimensions of the vector for fool,[36,58,1,4], correspond to the four Shakespeare plays.
- The same four dimensions are used to form the vectors for the other 3 words.
- Similar words have similar vectors because they tend to occur in similar documents.
- The term-document matrix lets us represent the meaning of a word by the documents it tends to occur in.

- It is most common to use a different kind of context for the dimensions of a word’s vector representation.
- Rather than the term-document matrix we use term-term matrix.
- Term-term matrix is more commonly called word-word matrix or term-context matrix.
- The columns of the term-term matrix are labeled by words rather than documents
- This matrix is thus of dimensionality |V| x |V|.
- Each cell records the number of times the row word and the column word co-occur in some context.
- We can create a window around the word, for example of 4 words to the left and 4 words to the right.
- The cell represents the number of times the column word occurs in a ±4 word window around the row word

- Here are 7-word windows surrounding four sample words from the Brown corpus:



sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of,

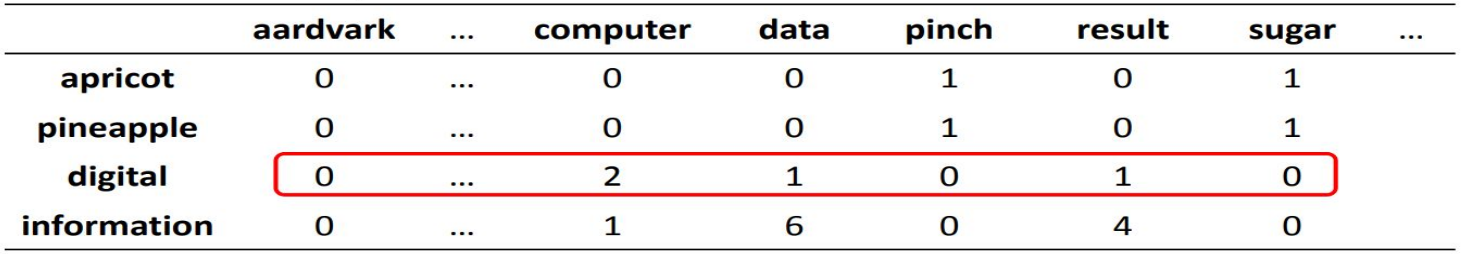
their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened

well suited to programming on the digital **computer**. In finding the optimal R-stage policy from

for the purpose of gathering data and **information** necessary for the study authorized in the

- For each word we collect the counts of the occurrences of context words.

- The following table shows a selection from the word- word co-occurrence matrix computed from the Brown corpus for these four words.



	aardvark	...	computer	data	pinch	result	sugar	...
apricot	0	...	0	0	1	0	1	
pineapple	0	...	0	0	1	0	1	
digital	0	...	2	1	0	1	0	
information	0	...	1	6	0	4	0	

Co-occurrence vectors for four words, computed from the Brown corpus, showing only six of the dimensions (hand-picked for pedagogical purposes). The vector for the word *digital* is outlined. Note that a real vector would have vastly more dimensions and thus be much sparser.

- The two words apricot and pineapple are more similar to each other than they are to other words like digital.
- Conversely, digital and information are more similar to each other than they are to other words like apricot.

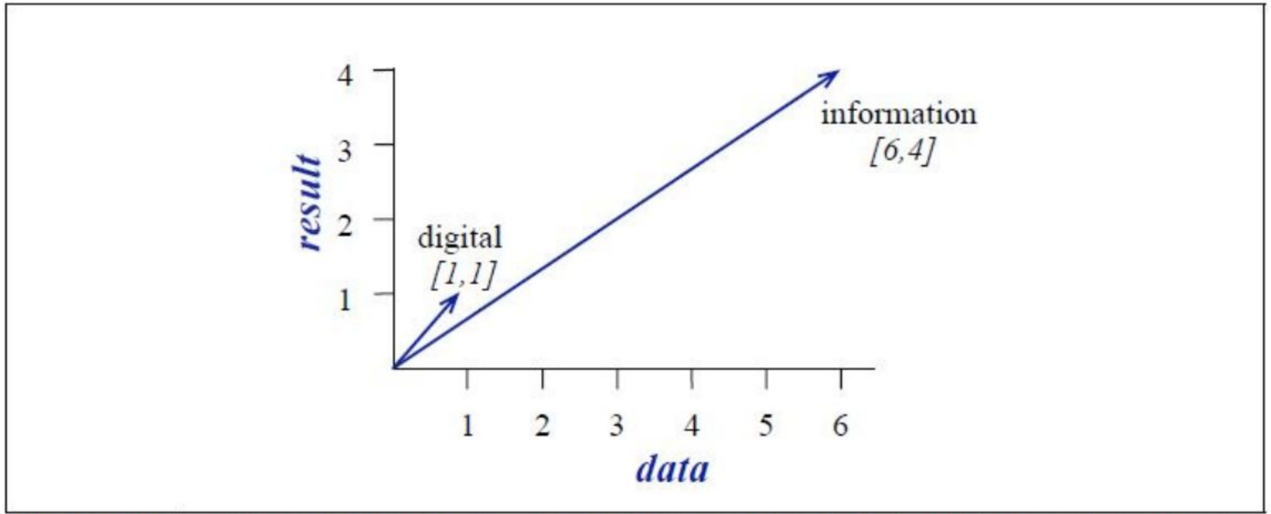


Figure 15.5 A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *result*.

- The size of the window used to collect counts can vary based on the goals of the representation.
- Generally, the size of the window is between 1 and 8 words on each side of the target word.

- TF-IDF (Term Frequency-Inverse Document Frequency) is a widely used statistical measure in text processing and information retrieval to evaluate the importance of a word or term within a document relative to a collection of documents (corpus).
- It helps in identifying the most significant terms in a document while reducing the impact of common words that occur frequently across many documents.

Key Components

1. Term Frequency (TF):

Definition: Measures how frequently a term appears in a document.

Formula: $\mathrm{TF}(t, d) = \dfrac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$

Purpose: Captures the local importance of a term within a specific document. Higher frequency indicates higher importance within that document.

2. Inverse Document Frequency (IDF):

Definition: Measures how important a term is by considering its occurrence across all documents in the corpus. The idea is that terms appearing in many documents are less informative than those appearing in fewer documents.

Formula: $\mathrm{IDF}(t, D) = \log \frac{\text{Total number of documents } D}{\text{Number of documents containing term } t}$

Purpose: Penalizes common terms and highlights terms that are unique to a few documents.

3. TF-IDF Score:

Definition: Combines TF and IDF to determine the importance of a term in a document relative to the entire corpus.

Formula: $\text{TF-IDF}(t, d, D) = \mathrm{TF}(t, d) \times \mathrm{IDF}(t, D)$

Purpose: Balances term frequency with the inverse document frequency to emphasize terms that are frequent in a document but not across the corpus, thus identifying terms that are important and specific to the document

Measuring Similarity

- Measuring similarity between words or concepts is a key task in natural language processing and computational linguistics.
- Various methods can be used to determine how similar two words or concepts are, depending on the context and the type of data available.
- Here’s a broad overview of different approaches:

1) Distributional Similarity is a technique used to measure how similar two words or phrases are based on their distributional properties in a large corpus of text. The core idea is that words that appear in similar contexts tend to have similar meanings.

Core Principles

- Contextual Similarity: Words that appear in similar contexts are likely to have similar meanings. For example, "cat" and "dog" often appear in similar contexts related to pets or animals.
- Vector Space Models: Distributional similarity is often implemented using vector space models, where words are represented as vectors in a high-dimensional space. The similarity between words can be measured by comparing these vectors.

1. Cosine Similarity

- Cosine similarity measures the cosine of the angle between two vectors in the embedding space. It is computed as:

$$\text{Cosine Similarity}(u, v) = \frac{u \cdot v}{||u||\,||v||}$$

- Where *u* and *v* are vectors representing the words, and // · // denotes the vector norm. Values close to 1 indicate high similarity, while values close to 0 indicate low similarity.

2. Euclidean Distance

- Euclidean distance measures the straight-line distance between two vectors in the embedding space. It is computed as:

$$\text{Euclidean Distance}(u, v) = \sqrt{\sum (u_i - v_i)^2}$$

- Where *ui* and *vi* are the components of the vectors. Smaller distances indicate higher similarity.

3. Word Embeddings

- **Word2Vec:** Developed by Google, this model creates word vectors by training on large corpora. It can use either the Continuous Bag of Words (CBOW) model or the Skip-gram model. The resulting vectors capture semantic relationships between words. For instance, the vectors for "king" and "queen" are often close to each other in the embedding space.
- **GloVe (Global Vectors for Word Representation):** Developed by Stanford, GloVe creates word vectors by analyzing word co-occurrence statistics across the entire corpus. It combines local context windows with global statistics to produce embeddings.
- **FastText:** An extension of Word2Vec developed by Facebook, FastText takes into account subword information (e.g., character n-grams), which helps in handling rare words and morphological variations better.

4. Contextual Embeddings

- **BERT (Bidirectional Encoder Representations from Transformers):** BERT provides embeddings for words based on their context in a sentence. Unlike static word embeddings, BERT’s embeddings change depending on the surrounding words, capturing nuanced meanings.
- **GPT (Generative Pre-trained Transformer):** Similar to BERT, GPT generates contextual embeddings for words or phrases. It uses the transformer architecture to consider the full context of words in a sentence.

Applications

- Semantic Similarity: Measuring how semantically similar two words or phrases are.
- Word Sense Disambiguation: Determining the correct meaning of a word based on its context.
- Text Classification: Using word vectors to classify text into categories.
- Information Retrieval: Enhancing search engines by finding documents similar to a query.
- Machine Translation: Improving translation quality by understanding word similarities across languages.

Advantages:

- Captures subtle semantic relationships between words.
- Can be trained on large corpora, making it applicable to diverse text sources.

Limitations:

- Static embeddings (e.g., Word2Vec, GloVe) may not capture context-dependent meanings.
- Requires large amounts of text data for training.
- Embeddings may not handle out-of-vocabulary words or rare words well (though FastText and subword approaches help mitigate this).

2. Semantic Similarity

1. **WordNet-Based Similarity:** WordNet provides several ways to measure semantic similarity:
2. **Path Similarity:** Measures similarity based on the shortest path between synsets in the WordNet hierarchy. For example, if two words share a common ancestor in the hyponym hierarchy, they are considered similar.
3. **Wu-Palmer Similarity:** Computes similarity based on the depth of the two synsets in the WordNet hierarchy and their lowest common ancestor. This method calculates similarity as a function of their common ancestor's depth.
4. **Leacock-Chodorow Similarity:** Considers the shortest path between two synsets and the maximum depth of the WordNet taxonomy. It normalizes the path length by the maximum depth.
5. **Resnik Similarity:** Uses information content to measure similarity based on the shared information content of the lowest common ancestor.

3. Contextual Similarity

1. **Contextual Embeddings:** Modern models like BERT and GPT generate context-dependent embeddings for words, capturing their meanings in specific contexts. Similarity can be assessed by comparing these embeddings using cosine similarity or other distance measures.
2. **Sentence Embeddings:** Tools like Sentence-BERT provide embeddings for entire sentences or phrases, allowing for the comparison of longer text segments beyond individual words.

4. Feature-Based Similarity

1. **Jaccard Similarity:** Measures similarity between sets of features (e.g., words or phrases) by comparing the intersection and union of these sets. It's often used in text comparison tasks.
2. **Overlap Coefficient:** Similar to Jaccard but focuses more on the intersection of feature sets rather than the union.

5. Knowledge-Based Similarity

1. **Taxonomic Similarity:** Uses hierarchical structures like ontologies or taxonomies (e.g., hierarchical classification of animals) to determine similarity based on the distance or relationship between concepts within these structures.
2. **Expert Annotations:** Sometimes similarity can be assessed based on human judgments or expert annotations, where a dataset of similarity ratings is used to train models.

Semantics with dense vectors

What is a Dense Vector?

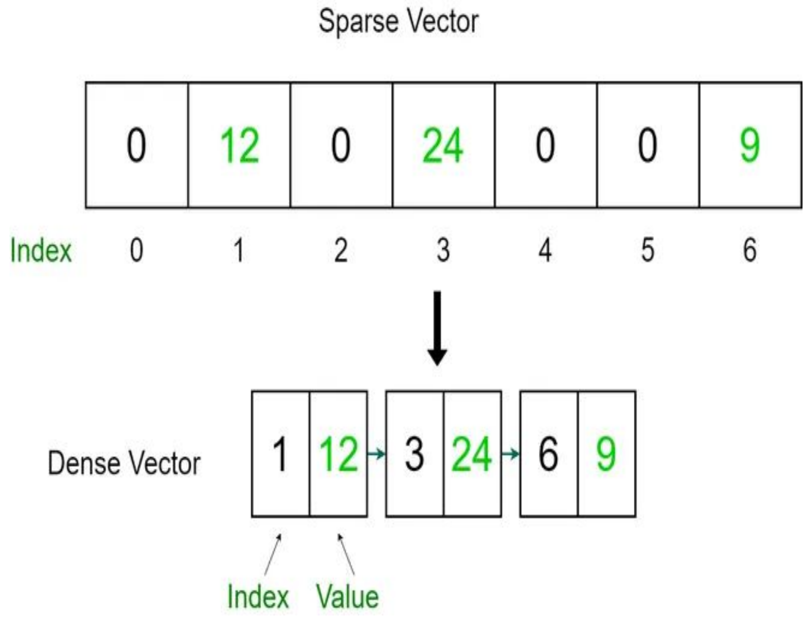
- Dense vectors are a type of mathematical objects that represent data in machine learning and artificial intelligence.
- They amount to a kind of vector, where non-zero values mainly populate its elements (features), opposite to sparse vectors overwhelmed with zeroes.
- Such representation allows the creating of models capable of capturing rich and subtle aspects characteristic of the data.
- Many machine learning applications require the use of dense vectors mainly due to their ability to describe subtle relations and nuances that may exist between data.
- In addition to this semantic similarity that cannot be seen in sparse vectors, other significant utilities of dense vectors include applying them to illustrate word frequency characteristics sets or packages for remotely supervised purposes such as solving or processing natural language, recommendation systems, and sentiment analysis.

Characteristics of Dense Vectors

- **Continuous Values:** Dense vectors are represented by real-valued numbers as opposed to the binary or categorical representation. Each dimension represents a feature or characteristic, and the value across that dimension is an actual number.
- **Semantic Richness:** Dense vectors are thus designed to capture semantic relationships that exist between entities. Hence, the dense vector representations indicate the similarity of objects or concepts. Therefore, when two things are similar, their vector representations are close in the vector space. This property allows meaning operations, like vector addition and subtraction, to reflect semantic relationships (e.g., “king” — “man” + “woman” results in a vector close to “queen”).
- **High Dimensionality:** Typically, their dimensionality is hundreds if not thousands in the case of dense vectors, where each dimension of the vector is usually representative of some feature or aspect of the data. Their high dimensionality results in active spaces that allow the encoding of much more information than sparse vectors like complex and nuanced relations.
- **Multi-Dimensional:** Dense vectors have more than one-dimensionality or length, making them multi-dimensional. A dense vector’s expressiveness and ability to depict complex relationships are strongly influenced by its number of dimensions.
- **Efficient Representations:** For some tasks, particularly those involving machine learning models where computing performance is critical, dense vectors are frequently more efficient than sparse representations.

Sparse Vectors vs Dense Vectors

- Both sparse and dense vectors are useful in many machine learning and artificial intelligence applications.
- While they have some characteristics, their representation and uses are not the same.



Significantly Representation

- **Sparse vectors:** mainly consisting of zeros, with a small number of non-zero elements denoting particular characteristics. Imagine them like wall-mounted light switches that can only have a few buttons on at once.
- **Dense vectors:** in most dimensions, there are non-zero values, which can be interpreted as greater complexity and precision to when it comes to data awareness. Dense vectors are like switches or dimmers — every level means different levels of information.
- **Sparse vectors:** Focus on explicitly represented features, explicitly stating which elements are present or absent.
- **Dense vectors:** Go beyond explicitly stated features. They capture complex relationships and subtle information within the data through the values and relationships between different dimensions.

Computational efficiency

- **Sparse vectors:** Require specialized algorithms for operations like distance and similarity calculations due to their sparsity.
- **Dense vectors:** Allow for efficient computation using standard vector operations due to their dense representation. This makes them faster and more convenient to work with applications.
- **Sparse vectors:** Ideal for high-dimensional data with few active features, such as text data with limited vocabulary or genetic data with few active genes.
- **Dense vectors:** More versatile and widely used in various applications like natural language processing, computer vision, recommendation systems, and anomaly detection.

Choosing between sparse and dense vectors:

The choice between sparse and dense vectors depends on the specific task and data characteristics. Consider the following factors:

- **Data size and sparsity:** If the data is high-dimensional with few active features, sparse vectors may be more efficient.
- **Computational requirements:** If computational efficiency is crucial, dense vectors might be a better choice.
- **Information richness and task complexity:** Dense vectors are preferred for tasks requiring capturing complex relationships and subtle information.

Methods to Create Dense Vectors

There are several ways to create dense vectors, and which one to choose depends on the requirements, data, and particular task at hand. These are a few common techniques for creating dense vectors.

- **Neural Language Models (NLMs):** Robust algorithms examine huge amounts of data to discover word, sentence, and document representations. A few well-known NLMs are Universal Sentence Encoder (USE), GloVe, and Word2Vec.
- **Word Embeddings:** Create dense vectors for individual words based on their relationships in text data.
- **Sentence Embeddings:** Generate dense vectors for entire sentences, capturing their overall meaning and relationships.
- **Vision Embeddings:** Using the visual characteristics of the photos and videos, extract dense representations.

Neural Language Models for Embeddings:

There are different ways to train neural language models and obtain embeddings, but two of the most popular ones are skip-gram and CBOW (Continuous Bag-of-Words). These methods are based on the idea that words that appear in similar contexts tend to have similar meanings, and therefore, similar embeddings.

- **Skip-gram:** Skip-gram is a method that takes a word as input and tries to predict the surrounding words in a window of a given size. For example, given the word "dog", skip-gram might try to predict "the", "brown", "barked", and "loudly" in a window of size 2. The embeddings are learned by optimizing the model to make accurate predictions.
- **CBOW (Continuous Bag-of-Words):** CBOW is a method that takes a group of words as input and tries to predict the word in the middle. For example, given the words "the", "brown", "_", "barked", and "loudly", CBOW might try to predict "dog" in the blank. The embeddings are learned by optimizing the model to make accurate predictions.

Skip-gram

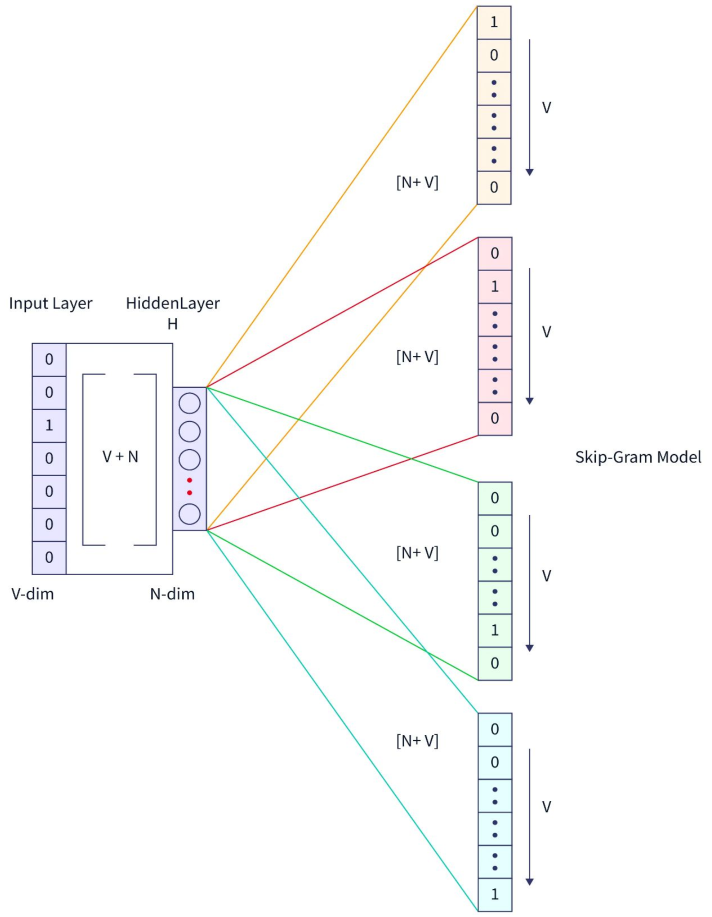
- The skip-gram model is a way of teaching a computer to understand the meaning of words based on the context they are used in.
- An example would be training a computer to understand the word "dog" by looking at sentences where "dog" appears and seeing the words that come before and after it.
- By doing this, the computer will be able to understand how the word "dog" is commonly used and will be able to use that understanding in other ways.

Here's an example to illustrate the concept:

- Let's say you have the sentence: **The dog fetched the ball.**
- If you are trying to train a skip-gram model for the word "dog", the goal of the model is to predict the context words "the" and "fetched" given the input word "dog". So, the training data for the model would be pairs of the form (input word = "dog", context word = "the"), (input word = "dog", context word = "fetched").

Architecture of Skip-Gram Model

- The architecture of the skip-gram model consists of an input layer, an output layer, and a hidden layer.
- The input layer is the word to be predicted, and the output layer is the context words.
- The hidden layer represents the embedding of the input word learned during training.
- The skip-gram model uses a feedforward neural network with a single hidden layer, as shown in the diagram
- The input and output layers are connected to the hidden layer through weights, adjusted during training to minimize the prediction error. The skip-gram model uses a negative sampling objective function to optimize the weights and learn the embeddings.



Let’s consider the sentence “She is a great dancer” and use a window size of 2. For the target word “a”, the context words are “She”, “is”, “great”, and “dancer”.

For Target Word "a":

Context Words: “She”, “is”, “great”, “dancer”

Pairs: (a, She), (a, is), (a, great), (a, dancer)

Each pair represents a training example where the model tries to predict the context words “She”, “is”, “great”, and “dancer” from the target word “a”.

In essence, the Skip-gram model:

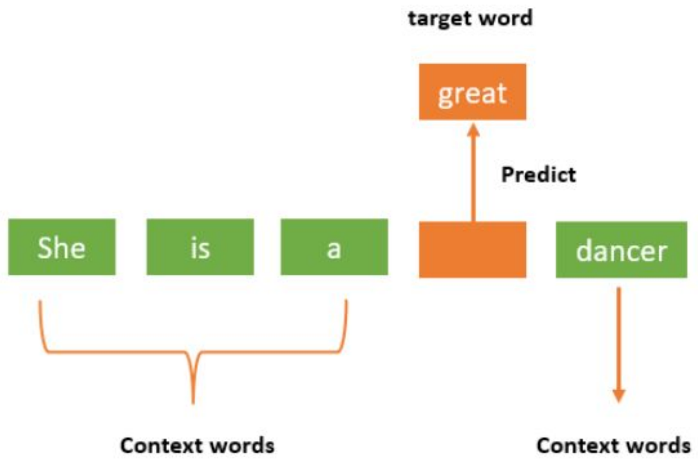
- Takes a target word as input.
- Predicts the surrounding context words within a fixed-size window.
- Learns word embeddings by optimizing the probability of the context words given the target word.

The Skip-gram model is particularly effective for learning word representations from large text corpora and can capture intricate relationships and semantics between words.

Continuous Bag of words

Continuous Bag of Words (CBOW) is a popular natural language processing technique used to generate word embeddings. Word embeddings are important for many NLP tasks because they capture semantic and syntactic relationships between words in a language.

CBOW is a neural network-based algorithm that predicts a target word given its surrounding context words. It is a type of “unsupervised” learning, meaning that it can learn from unlabeled data, and it is often used to pre-train word embeddings that can be used for various NLP tasks such as sentiment analysis, text classification, and machine translation.

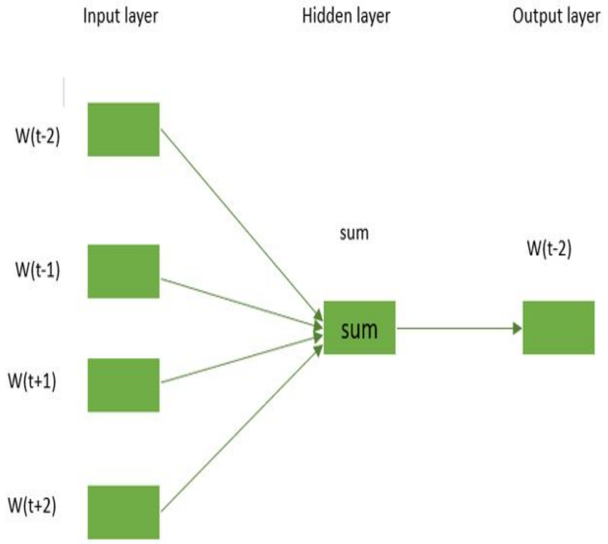


Difference between Bag-of-Words (BoW) model and the Continuous Bag-of-Words (CBOW)

- The Bag-of-Words model and the Continuous Bag-of-Words model are both techniques used in natural language processing to represent text in a computer-readable format, but they differ in how they capture context.
- The BoW model represents text as a collection of words and their frequency in a given document or corpus. It does not consider the order or context in which the words appear, and therefore, it may not capture the full meaning of the text. The BoW model is simple and easy to implement, but it has limitations in capturing the meaning of language.
- In contrast, the CBOW model is a neural network-based approach that captures the context of words. It learns to predict the target word based on the words that appear before and after it in a given context window. By considering the surrounding words, the CBOW model can better capture the meaning of a word in a given context.

The CBOW model uses the target word around the context word in order to predict it. Consider the above example “She is a great dancer.” The CBOW model converts this phrase into pairs of context words and target words. The word pairings would appear like this ([she, a], is), ([is, great], a) ([a, dancer], great) having window size=2.

The model considers the context words and tries to predict the target term. The four 1∗W input vectors will be passed to the input layer if have four words as context words are used to predict one target word. The hidden layer will receive the input vectors and then multiply them by a W∗N matrix. The 1∗N output from the hidden layer finally enters the sum layer, where the vectors are element-wise summed before a final activation is carried out and the output is obtained from the output layer.



Concept of Word Sense Disambiguation

Word Sense Disambiguation (WSD) is a subtask of Natural Language Processing that deals with the problem of identifying the correct sense of a word in context.

Many words in natural language have multiple meanings, and WSD aims to disambiguate the correct sense of a word in a particular context.

For example, the word “bank” can have different meanings in the sentences “I deposited money in the bank” and “The boat went down the river bank”.

WSD is a challenging task because it requires understanding the context in which the word is used and the different senses in which the word can be used. Some common approaches to WSD include:

1. **Supervised learning:** This involves training a machine learning model on a dataset of annotated examples, where each example contains a target word and its sense in a particular context. The model then learns to predict the correct sense of the target word in new contexts.
2. **Unsupervised learning:** This involves clustering words that appear in similar contexts together, and then assigning senses to the resulting clusters. This approach does not require annotated data, but it is less accurate than supervised learning.
3. **Knowledge-based:** This involves using a knowledge base, such as a dictionary or ontology, to map words to their different senses. This approach relies on the availability and accuracy of the knowledge base.
4. **Hybrid:** This involves combining multiple approaches, such as supervised and knowledge-based methods, to improve accuracy.

Difficulties in Word Sense Disambiguation

There are some difficulties faced by Word Sense Disambiguation (WSD).

1. **Different Text-Corpus or Dictionary:** One issue with word sense disambiguation is determining what the senses are because different dictionaries and thesauruses divide words into distinct senses. Some academics have proposed employing a specific lexicon and its set of senses to address this problem. In general, however, research findings based on broad sense distinctions have outperformed those based on limited ones. The majority of researchers are still working on fine-grained WSD.
2. **PoS Tagging:** Part-of-speech tagging and sense tagging have been shown to be very tightly coupled in any real test, with each potentially constraining the other. Both disambiguating and tagging with words are involved in WSM part-of-speech tagging. However, algorithms designed for one do not always work well for the other, owing to the fact that a word’s part of speech is mostly decided by the one to three words immediately adjacent to it, whereas a word’s sense can be determined by words further away.

Approaches for Word Sense Disambiguation

1. Supervised: The assumption behind supervised approaches is that the context can supply enough evidence to disambiguate words on its own (hence, world knowledge and reasoning are deemed unnecessary).

Supervised methods for Word Sense Disambiguation (WSD) involve training a model using a labeled dataset of word senses. The model is then used to disambiguate the sense of a target word in new text. Some common techniques used in supervised WSD include:

- **Decision list:** A decision list is a set of rules that are used to assign a sense to a target word based on the context in which it appears.
- **Neural Network:** Neural networks such as feedforward networks, recurrent neural networks, and transformer networks are used to model the context-sense relationship.

- **Support Vector Machines:** SVM is a supervised machine learning algorithm used for classification and regression analysis.
- **Naive Bayes:** Naive Bayes is a probabilistic algorithm that uses Bayes’ theorem to classify text into predefined categories.
- **Decision Trees:** Decision Trees are a flowchart-like structure in which an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.
- **Random Forest:** Random Forest is an ensemble learning method for classification, regression, and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes.

2. Unsupervised: The underlying assumption is that similar senses occur in similar contexts, and thus senses can be induced from the text by clustering word occurrences using some measure of similarity of context.

Using fixed-size dense vectors (word embeddings) to represent words in context has become one of the most fundamental blocks in several NLP systems. Traditional word embedding approaches can still be utilized to improve WSD, despite the fact that they conflate words with many meanings into a single vector representation.

Lexical databases (e.g., WordNet, ConceptNet, BabelNet) can also help unsupervised systems map words and their senses as dictionaries, in addition to word embedding techniques.

3. Knowledge-Based: It is built on the idea that words used in a text are related to one another, and that this relationship can be seen in the definitions of the words and their meanings.

The pair of dictionary senses having the highest word overlap in their dictionary meanings are used to disambiguate two (or more) words. Lesk Algorithm is the classical algorithm based on Knowledge-Based WSD.

Lesk algorithm assumes that words in a given “neighborhood” (a portion of text) will have a similar theme.

The dictionary definition of an uncertain word is compared to the terms in its neighborhood in a simplified version of the Lesk algorithm.

Introduction to WordNet

WordNet is a large lexical database of English designed to be a dictionary and thesaurus combined. It was created at Princeton University and is widely used in Natural Language Processing (NLP) and computational linguistics. WordNet groups words into sets of synonyms called "synsets," provides short definitions, and records various semantic relations between these synonym sets.

Key Concepts of WordNet:

1) Synsets (Synonym Sets):

- A synset is a group of words that are synonyms of each other in a specific context.
- Each synset is associated with a definition or gloss that provides the meaning of the words in that set.
- For example, the words "car," "automobile," "motorcar," and "machine" might belong to the same synset, as they refer to the same concept.

2) Semantic Relations:

- Hypernyms and Hyponyms: Hypernyms are more general terms (e.g., "vehicle" is a hypernym of "car"), while hyponyms are more specific (e.g., "sedan" is a hyponym of "car").
- Meronyms and Holonyms: Meronyms represent parts of a whole (e.g., "wheel" is a meronym of "car"), while holonyms are the whole (e.g., "car" is a holonym of "wheel").
- Antonyms: Pairs of words with opposite meanings (e.g., "hot" and "cold").
- Troponyms: Specific manners of doing something (e.g., "to stroll" is a troponym of "to walk").

3) Lexical Relations:

- Synonymy: The relation between words with similar meanings.
- Antonymy: The relation between words with opposite meanings.
- Polysemy: The phenomenon where a single word has multiple meanings (e.g., the word "bank" can refer to a financial institution or the side of a river).
- Homonymy: Words that sound the same but have different meanings (e.g., "bat" can refer to a flying mammal or a piece of sports equipment).

4) Part of Speech:

- WordNet organizes words by their part of speech, such as nouns, verbs, adjectives, and adverbs.
- Each part of speech has its own set of synsets and semantic relations.

Applications:

- Word Sense Disambiguation: WordNet helps in determining which sense of a word is being used in a context.
- Information Retrieval: Enhances search engines by expanding queries with synonyms and related terms.
- Text Analysis: Assists in sentiment analysis, text classification, and other NLP tasks by providing a structured understanding of word meanings.
- Machine Translation: Helps in translating words and phrases by providing synonyms and context-specific meanings.

Structure:

- WordNet is structured like a thesaurus, but it goes beyond simple synonyms and antonyms to provide a richer and more detailed representation of the relationships between words.
- The database is hierarchical, with more specific concepts branching out from more general ones.

NLP Applications and Case Studies

Intelligent Work Processors: Machine Translation

- Machine Translation (MT) has become an essential tool in our increasingly globalized world, enabling real-time communication and understanding across different languages.
- Intelligent work processors in the realm of machine translation leverage advanced technologies like artificial intelligence and deep learning to provide accurate, context-aware, and efficient translation services.
- Machine Translation involves the use of software to translate text or speech from one language to another without human intervention.
- Over the years, MT has evolved from simple rule-based systems to complex neural networks capable of understanding context and nuance.

Key Applications:

- **Global Communication:** Facilitating conversations and collaborations across different languages.
- **Content Localization:** Adapting products, services, and content to fit local cultures and languages.
- **Real-Time Translation:** Enabling instant translation in applications like live chats, customer support, and travel assistance.
- **Accessibility:** Making information accessible to non-native speakers and those with language barriers.

Components of Intelligent Machine Translation Processors

1. Advanced Neural Networks

- Deep Learning Models: Utilize large datasets to learn and predict translations, capturing complex language structures and contextual meanings.
- Transformer Architectures: Models like Google's BERT and OpenAI's GPT excel in understanding context, enabling more accurate and fluent translations.
- Sequence-to-Sequence Learning: Processes entire sentences or paragraphs, maintaining coherence and contextual relevance.

2. Contextual Understanding

- Semantic Analysis: Interprets the meaning behind words and phrases to provide translations that preserve intent.
- Disambiguation Techniques: Resolves ambiguities in language by considering surrounding text and probable meanings.
- Cultural Nuance Recognition: Adapts translations to fit cultural contexts, idioms, and colloquialisms appropriately.

3. Real-Time Processing

- Low Latency Responses: Provides instantaneous translations, crucial for live conversations and time-sensitive communications.
- Edge Computing Integration: Processes data locally on devices to reduce reliance on internet connectivity and enhance speed.
- Scalability: Handles large volumes of data and multiple concurrent translation requests efficiently.

4. Multimodal Translation

- Text-to-Speech and Speech-to-Text: Converts spoken language to text and vice versa, supporting diverse communication forms.
- Image and Video Translation: Extracts and translates text from visual media using OCR (Optical Character Recognition) and video processing technologies.
- Gesture and Sign Language Translation: Emerging technologies aim to interpret and translate non-verbal communication forms.

5. User Feedback and Continuous Learning

- Interactive Correction Mechanisms: Allows users to correct translations, enabling the system to learn and improve over time.
- Adaptive Learning: Continuously updates models based on new data and usage patterns to enhance accuracy.
- Quality Assessment Metrics: Implements evaluation frameworks to monitor and maintain translation standards.

6. Security and Privacy

- Data Encryption: Ensures that sensitive information remains secure during translation processes.
- On-Device Processing: Minimizes data exposure by performing translations locally when possible.
- Compliance with Regulations: Adheres to international data protection laws like GDPR to maintain user trust.

7. Integration and Accessibility

- API Support: Allows easy integration with various applications, websites, and services.
- Cross-Platform Compatibility: Supports multiple devices and operating systems for widespread accessibility.
- User-Friendly Interfaces: Provides intuitive interfaces for both end-users and developers to interact with translation services seamlessly.

Intelligent Work Processors: User Interfaces

Intelligent Work Processors: User Interfaces refer to user interfaces designed to enhance productivity through intelligent automation, AI, or advanced data processing. These interfaces are crucial in modern workflows, where efficiency, ease of use, and adaptability are essential.

Key Aspects of Intelligent Work Processor UIs:

1) Automation and AI Integration:

- Context-Aware Assistance: Interfaces that provide real-time suggestions, auto-completions, or contextual help based on the user's activity.
- Task Automation: Automated workflows that reduce manual input, such as automatically categorizing emails or generating reports.

Adaptive User Experience:

- Personalization: UIs that adapt to individual user preferences, optimizing the layout, shortcuts, and tools based on usage patterns.
- Dynamic Interfaces: Components that change based on the task at hand, streamlining the user experience.

Collaboration and Connectivity:

- Integrated Communication Tools: Embedding messaging, video conferencing, and collaborative document editing within the workflow.
- Cross-Platform Consistency: Ensuring a seamless experience across devices, from desktops to mobile devices.

Data-Driven Decision Support:

- Real-Time Analytics: Dashboards and data visualization tools that provide actionable insights.
- Predictive Models: Interfaces that offer predictions or recommendations based on historical data.

User-Centric Design:

- Intuitive Navigation: Simple and clear pathways to essential tools and features, reducing the learning curve.
- Accessibility: Ensuring that the interface is usable by people with varying abilities, including those who rely on screen readers or alternative input devices.

These intelligent interfaces are designed to make work more efficient and effective by leveraging advanced technologies while maintaining a user-friendly experience.

Speech Recognition

Speech Recognition, also known as automatic speech recognition (ASR), computer speech recognition, or speech-to-text, focuses on enabling computers to understand and interpret human speech.

Speech recognition involves converting spoken language into text or executing commands based on the recognized words.

This technology relies on sophisticated algorithms and machine learning models to process and understand human speech in real-time, despite the variations in accents, pitch, speed, and slang.

Key Features of Speech Recognition

1. **Accuracy and Speed:** They can process speech in real-time or near real-time, providing quick responses to user inputs.
2. **Natural Language Understanding (NLU):** NLU enables systems to handle complex commands and queries, making technology more intuitive and user-friendly.
3. **Multi-Language Support:** Support for multiple languages and dialects, allowing users from different linguistic backgrounds to interact with technology in their native language.
4. **Background Noise Handling:** This feature is crucial for voice-activated systems used in public or outdoor settings.

Speech Recognition Algorithms

Speech recognition technology relies on complex algorithms to translate spoken language into text or commands that computers can understand and act upon. Here are the algorithms and approaches used in speech recognition:

1. Hidden Markov Models (HMM)

Hidden Markov Models have been the backbone of speech recognition for many years. They model speech as a sequence of states, with each state representing a phoneme (basic unit of sound) or group of phonemes. HMMs are used to estimate the probability of a given sequence of sounds, making it possible to determine the most likely words spoken. Usage: Although newer methods have surpassed HMM in performance, it remains a fundamental concept in speech recognition, often used in combination with other techniques.

2. Natural language processing (NLP)

NLP is the area of artificial intelligence which focuses on the interaction between humans and machines through language through speech and text. Many mobile devices incorporate speech recognition into their systems to conduct voice search. Example such as: Siri or provide more accessibility around texting.

3. Deep Neural Networks (DNN)

DNNs have improved speech recognition’s accuracy a lot. These networks can learn hierarchical representations of data, making them particularly effective at modeling complex patterns like those found in human speech. DNNs are used both for acoustic modeling, to better understand the sound of speech, and for language modeling, to predict the likelihood of certain word sequences.

4. End-to-End Deep Learning

Now, the trend has shifted towards end-to-end deep learning models, which can directly map speech inputs to text outputs without the need for intermediate phonetic representations. These models, often based on advanced RNNs, Transformers, or Attention Mechanisms, can learn more complex patterns and dependencies in the speech signal.

Uses of Speech Recognition

Virtual Assistants: These are like digital helpers that understand what you say. They can do things like set reminders, search the internet, and control smart home devices, all without you having to touch anything. Examples include Siri, Alexa, and Google Assistant.

1. **Accessibility Tools:** Speech recognition makes technology easier to use for people with disabilities. Features like voice control on phones and computers help them interact with devices more easily. There are also special apps for people with disabilities.

2. **Automotive Systems:** In cars, you can use your voice to control things like navigation and music. This helps drivers stay focused and safe on the road. Examples include voice-activated navigation systems in cars.

3. Healthcare: Doctors use speech recognition to quickly write down notes about patients, so they have more time to spend with them. There are also voice-controlled bots that help with patient care. For example, doctors use dictation tools to write down patient information quickly.

4. Customer Service: Speech recognition is used to direct customer calls to the right place or provide automated help. This makes things run smoother and keeps customers happy. Examples include call centers that you can talk to and customer service bots.

5. Education and E-Learning: Speech recognition helps people learn languages by giving them feedback on their pronunciation. It also transcribes lectures, making them easier to understand. Examples include language learning apps and lecture transcribing services.

6. Security and Authentication: Voice recognition, combined with biometrics, keeps things secure by making sure it’s really you accessing your stuff. This is used in banking and for secure facilities. For example, some banks use your voice to make sure it’s really you logging in.

7. Entertainment and Media: Voice recognition helps you find stuff to watch or listen to by just talking. This makes it easier to use things like TV and music services. There are also games you can play using just your voice.

Commercial use of NLP: NLP in customer Service

NLP is the basis for any “smart” service or AI that analyzes phone calls or messaging conversations for customer service teams.

It can be used in a variety of ways in customer service, from improving customer-facing interactions with real-time coaching for agents, to gathering customer insights, analyzing customer sentiment, and more.

Essentially, wherever there is spoken or written language in your customer service journey, NLP plays a role.

5 use cases of NLP in customer service

1. Analyzing customer sentiment

When supervisors are overseeing agents, they might have tens or even hundreds of live customer calls happening simultaneously. How can one person monitor everything?

One thing NLP can do is provide conversation analytics, such as by flagging sentiment in these calls based on the words being used in the conversations.

Whether it's a price objection, a satisfied customer, or a general lack of enthusiasm—NLP can pick up on certain words or phrases to understand the customer’s sentiment. If it’s positive, that’s great!

If a supervisor sees a call with negative sentiment, they can quickly pull up the real-time transcript to get more context before deciding if they need to listen in or barge the call to help de-escalate.

For example: If you get these responses from feedback forms:

“The agent I spoke to was awesome.”

“My order arrived quicker than I expected.”

“It’s easy to sync my data. Thanks for putting together your onboarding docs!”

The machine learning system will then tell you that the vast majority of feedback is positive. This gives you a rough understanding of how well you’re performing.

SENTIMENT ANALYSIS and EMOTION MINING

Sentiment analysis, sometimes referred to as opinion mining, is the process of analyzing digital text at scale to uncover the emotional tone—or sentiment—of the message.

This technique is a subset of AI that combines natural language processing (NLP), machine learning (ML) and data mining to accurately determine the thinking, attitude or emotions behind unstructured text data.

As video and audio data become more common, methods for extracting sentiment from these data types are also being developed.

Types of sentiment analysis

Sentiment analysis extends far beyond simply gauging how customers feel about a brand, product or service. This versatile technique can be used for everything from basic polarity detection to more advanced emotion and aspect-based analysis.

1. Intent-based sentiment analysis

Intent-based analysis is used to identify the underlying meaning contained within text, uncovering the intention behind a user's statement. Gathering information using this method may include categorizing the text as a question, statement, opinion or desire; detecting the sentiment it conveys; and ranking the intensity with which it is being expressed. For example, a company might evaluate employee feedback on recent changes to benefits or compensation and the intensity behind those responses.

2. Aspect-based sentiment analysis

Aspect-based sentiment analysis categorizes data by aspect, identifying the sentiment attached to a specific category, feature or topic. This is a more advanced form of sentiment analysis used to identify how customers feel about a certain aspect of a product or customer service experience. Examples include identifying how users felt about a newly added feature and gauging how effective a customer support representative was at resolving an issue.

3. Fine-grained sentiment analysis

As the name implies, fine-grained sentiment analysis extracts specific opinion elements from text. This form of sentiment analysis moves beyond more basic methods that classify sentiment simply as positive, negative or neutral. It divides text intent into multiple levels and incorporates more nuanced opinion elements to reveal user preferences as they relate to specific product aspects and corresponding emotional words. An application of this approach could include analyzing online news coverage to gauge how people perceive and react to specific events.

4. Emotion detection

Emotion detection involves identifying specific emotional states such as anger, fear, sadness and excitement. This approach may be used to monitor social media channels for customer reactions to new products or service launches.

Challenges to effective sentiment analysis

Written language and the meaning behind it is full of complexity. Teasing out contextual, cultural and linguistic nuances to arrive at an accurate conclusion can be a significant challenge. But these barriers can be overcome, including through these four methods.

1. Detecting negative text

Text that includes negation doesn’t necessarily convey a negative sentiment. A statement such as “This vacuum does not leave pet hair on the couch like my old one!” is actually a positive assessment of a product’s performance. But sentiment analysis algorithms can struggle to interpret text that contains negations accurately. Using large, annotated datasets that include a broad number of negation words or phrases during model training can improve the ML model’s ability to distinguish between negated and non-negated text.

2. Understanding context

Human language is filled with complexity. Sentiment analysis tools can struggle with text where the meaning is context-dependent. Irony and sarcasm are two examples that present significant challenges. Training models to use contextual clues such as linguistic markers, punctuation, emojis and hashtags to detect and interpret statements that contain irony or sarcasm can help them label content-dependent text with greater accuracy.

3. Accommodating language and cultural differences

Many organizations operate in multiple countries, making it essential to account for linguistic and cultural differences when conducting sentiment analysis. Many of today’s natural language processing systems provide support for multiple languages, streamlining the process of analyzing user content written in more than one language.

4. Translating emojis

Emojis pose a unique challenge for sentiment analysis tools that are designed to work with written text. These graphical representations of sentiment or emotion are often a primary means of expression for many customers. Ignoring them during text analysis can significantly decrease the accuracy of your result. Converting emojis to their textual description can improve sentiment classification accuracy and resolve out-of-vocabulary issues that occur when a model encounters information that was not part of the training data.

2. Conversational AI (BOTS)

Many people use “chatbots” and “conversational AI” interchangeably, but they’re technically two different things.

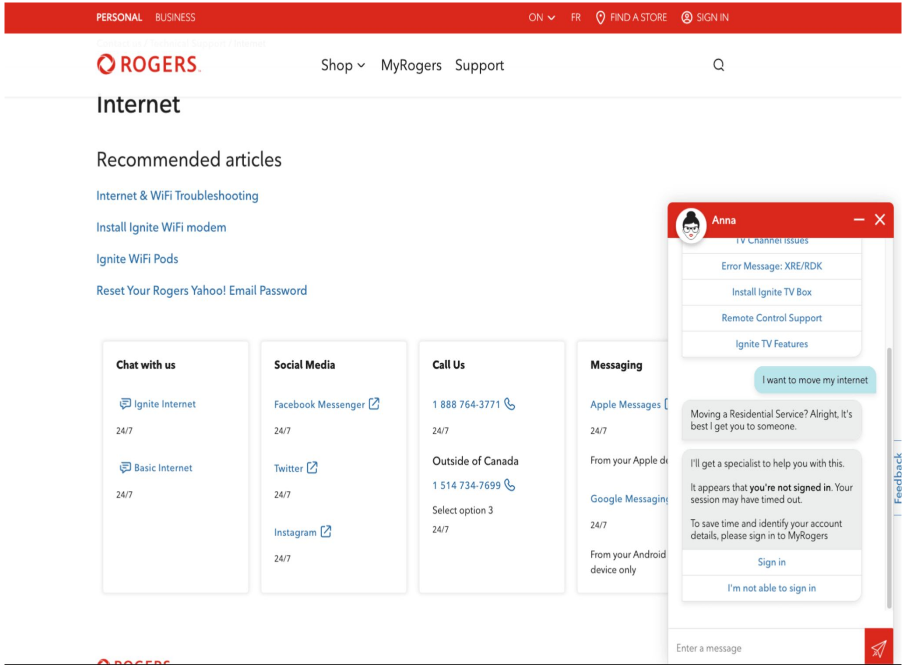
Yes, both chatbots and conversational AI do similar jobs—they’re usually placed on a website to answer customer questions automatically to remove some of the burden from live agents.

But conversational AI is different in that it’s a more advanced form of chatbot. Whereas chatbots typically can only answer preset questions with preset answers, conversational AI is much closer to being able to simulate a conversation with a customer.

For example, see how in this example on Rogers (a telecom and internet provider in Canada), the customer says they need to set up a move for their services instead of selecting one of the preset options, and the AI understood, thanks to NLP—and provided the correct answer options.

Conversational AI answering a question about moving internet services

A traditional chatbot would not be able to do that.



Internet

Recommended articles

Internet & WiFi Troubleshooting

Install Ignite WiFi modem

Ignite WiFi Pods

Reset Your Rogers Yahoo! Email Password

Chat with us

Ignite Internet

24/7

Basic Internet

24/7

Social Media

Facebook Messenger

24/7

Twitter

24/7

Instagram

24/7

Call Us

1 888 764-3771

24/7

Outside of Canada

1 514 734-7699

Select option 3

24/7

Messaging

Apple Messages

24/7

From your Apple de

Google Messaging

24/7

From your Android device only

Anna

TV Channel issues

Error Message: XRE/RDK

Install Ignite TV Box

Remote Control Support

Ignite TV Features

I want to move my internet

Moving a Residential Service? Alright, It's best I get you to someone.

I'll get a specialist to help you with this.

It appears that **you're not signed in**. Your session may have timed out.

To save time and identify your account details, please sign in to MyRogers

Sign in

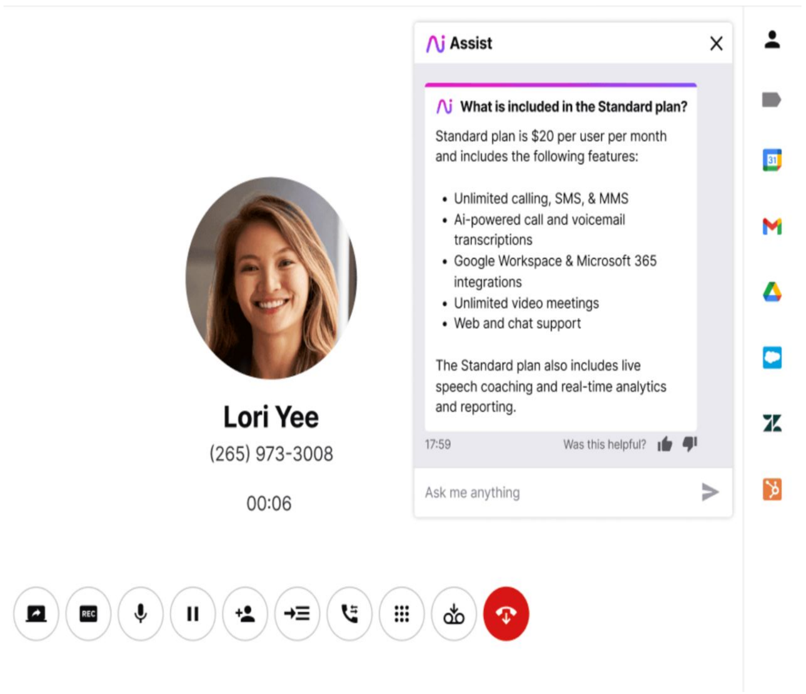
I'm not able to sign in

3. Live agent support

NLP doesn't just improve the customer experience, it can also help your customer support agents live during conversations.

We already mentioned that conversational AI can help deflect calls and reduce overwhelming volumes of inquiries for agents.

But beyond that, NLP can also play a part in AI-powered real-time assists for agents. For example, Dialpad's Ai Agent Assist can understand customer questions and automatically search all connected knowledge sources (even unstructured sources like PDFs and past customer conversations) to find helpful information for agents in real time:



What is included in the Standard plan?

Standard plan is $20 per user per month and includes the following features:

- Unlimited calling, SMS, & MMS
- Ai-powered call and voicemail transcriptions
- Google Workspace & Microsoft 365 integrations
- Unlimited video meetings
- Web and chat support

The Standard plan also includes live speech coaching and real-time analytics and reporting.

4. Improve training and knowledge base

On a related note, contact center AI can also use NLP to fill in gaps in training and knowledge bases.

This all depends on the robustness of the AI and NLP engine—some customer service software that claim to have AI may not be able to do this at all. But how it essentially works, at least with Dialpad Ai, is that the AI can flag where there’s missing content to supervisors when customers ask questions that aren’t covered in the current knowledge bases.

This automates a lot of that ongoing maintenance work when it comes to updating contact center training materials, and can save supervisors and enablement teams quite a bit of time.

5. Capturing conversation insights

NLP can also help with capturing insights from conversations.

Dialpad Ai, for example, can capture key moments and action items during calls and video meetings, which are automatically emailed along with a searchable transcript in a post-call summary to attendees:

This can be helpful for supervisors and agents who may need to double back and review certain customer calls. But alongside this, supervisors can also track how frequently certain topics come up in customer support conversations.

For example, if they want to see how often customers are calling in about refunds, they can create a “Custom Moment” in Dialpad to monitor how often “refund,” “cancel,” and/or “money back” is mentioned on a call:

You can track pretty much anything with Custom Moments—dates, email addresses, frequent support issues, product names—and yes, even curse words—can be tracked and set up with alerts.

But besides this, NLP can help with other post-call tasks as well. Because NLP processes language and conversations, it can even do things like suggest dispositions for calls to reduce after-call work (ACW) time for agents and help automate QA scoring, which saves time for supervisors.

Many people still only think of voice transcriptions when they hear “capturing conversational insights,” but the possibilities really are endless, and we’re only just starting to see what customer service teams can really do with NLP.

Handling Frauds and SMS

Steps to detect Frauds and SMS

1. First try to understand the data and its distribution with basic Exploratory Data Analysis with the help of Pandas and Matplotlib libraries. Also, check for any outliers by analysing the distribution graphs.

2. Now with the help of NLP library "NLTK", first remove the punctuation and special symbols from all the SMS and then lower case them. You can even tokenize each SMS into sentences and words after removing punctuation & special symbols. Here I am just splitting each SMS into words with white spaces. However, tokenization and parsing may be the best idea to split the texts. Please note that converting all the data to lower case helps in the process of preprocessing and in later stages in the NLP application.

3. Then remove the Stopswords from all the SMS.

4. After processing each SMS, we will create the WordCloud for Spam and Ham messages for the visual representation of widely used words in both Spam and Ham messages.

5. Now we can normalize the text by NLTK lemmatization or stemming or distinguishing by part of speech (POC). However, sometimes these methods don't work well especially for text-messages due to the way a lot of people tend to use abbreviations or shorthand in SMS. E.g. "IDK" for "I don't know" or "wut" for "what". So we will not process the text by these methods.

6. For now, we will have the messages as lists of tokens and now we need to convert each of these messages into a vector so that SciKit Learn's algorithm models can work with.

We'll do that in three steps using the bag-of-words (BOW) model:

* Count how many times does a word occur in each message (Known as term frequency - TF)

* Weigh the counts, so that frequent tokens get lower weight (inverse document frequency - IDF)

* Normalize the vectors to unit length, to abstract from the original text length (L2 norm)

7. Once the messages represented as vectors, we can finally train our spam/ham classifier. Now we can actually use almost any sort of classification algorithms like Random Forest, Naive Bayes etc.

LSTM models

Natural language processing (NLP) tasks frequently employ the Recurrent Neural Network (RNN) variant known as Long Short-Term Memory (LSTM). RNNs are neural networks that process sequential data, such as time series data or text written in a natural language. A particular kind of RNN called LSTMs can solve the issue of vanishing gradients, which arises when traditional RNNs are trained on lengthy data sequences.

A collection of “memory cells” that can store information and transmit it from one time step to the next makeup LSTMs. A system of “gates” that regulate data flow into and out of the cells connects these cells. The input gate, forget gate, and output gate are the three different types of gates that make up an LSTM.

The input gate governs the flow of new information into the cell, the forget gate regulates the flow of information out of the cell, and the output gate manages the data flow into the LSTM’s output. By controlling the flow of information in this way, LSTMs can forget information that isn’t important while remembering other information for longer.

LSTM has been used in many Natural Language Processing (NLP) tasks, such as:

- Language Modelling
- Text Generation
- Machine Translation
- Sentiment Analysis
- Text Classification

In NLP, LSTMs are typically trained to classify the overall meaning or sentiment of the text or to take in a sequence of words as input and predict the next word in the series. These NLP tasks are a good fit for LSTMs because they can handle sequential data well and keep track of previous inputs in “memory.”

LSTMs can retain information for a long time while forgetting irrelevant information.

Bidirectional LSTM (BiLSTM) are another LSTM variant that helps maintain the context of the past and future when making predictions.

Why use an LSTM in NLP tasks?

When used for natural language processing (NLP) tasks, Long Short-Term Memory (LSTM) networks have several advantages.

1. **Handling sequential data:** Since LSTMs are built to handle sequential data, they are ideal for NLP tasks like language modelling, machine translation, and text generation. In their concealed states, they can store information from the past and use it to forecast the future.

2. **Handling long-term dependencies:** When dealing with long-term dependencies in sequential data, LSTMs excel. They can better comprehend context and meaning in text because they can keep information hidden for extended periods.

3. **Handling missing data:** LSTMs are robust to errors and missing data because they can take missing data in the input sequence. This can be helpful when performing tasks like speech recognition, where the input may be noisy or lacking.

4. **Handling variable-length inputs:** In tasks like text classification, where the length of the input text may vary, LSTMs' ability to handle variable-length input sequences can be helpful.

5. **Handling a large amount of data:** The training process can be sped up by using parallel computing methods like GPUs and TPUs in combination with LSTMs, which can handle large amounts of data.

6. **Attention Mechanism:** When LSTM networks are combined with an attention mechanism, they can focus on specific parts of the input sequence. This helps with tasks like machine translation and summarising text.

7. **Combining LSTMs with other models:** Like the encoder-decoder model used for machine translation and the attention-based model used for text summarization, LSTMs can be combined with other models to make more powerful architectures.

How to implement an LSTM in NLP for text classification

Long Short-Term Memory (LSTM) can be effectively used for text classification tasks. In text classification, the goal is to assign one or more predefined categories or labels to a piece of text. LSTMs can be trained by treating each word in the text as a time step and training the LSTM to predict the label of the text.

First, the text needs to be transformed into a numerical representation, which can be accomplished by employing tokenization and word embedding strategies. Tokenization involves separating the text into its words, and word embedding, which requires mapping words to high-dimensional vectors that accurately capture their meaning, are two methods for doing this.

The LSTM would then be fed these numerical representations of the text. Each word in the sequence will be processed by the LSTM one at a time, producing a hidden state for each word. The label of the text can be predicted using these hidden states, which capture the meaning of the text up to that point.

To generate the class scores, the output of the LSTM is fed into a fully connected layer and a softmax activation function. The class scores will represent the probability distribution of each possible class. The final predicted class is the one with the highest probability.

BERT models

BERT (Bidirectional Encoder Representations from Transformers) leverages a transformer-based neural network to understand and generate human-like language. BERT employs an encoder-only architecture. In the original Transformer architecture, there are both encoder and decoder modules. The decision to use an encoder-only architecture in BERT suggests a primary emphasis on understanding input sequences rather than generating output sequences.

Bidirectional Approach of BERT :

Traditional language models process text sequentially, either from left to right or right to left. This method limits the model’s awareness to the immediate context preceding the target word. BERT uses a bi-directional approach considering both the left and right context of words in a sentence, instead of analyzing the text sequentially, BERT looks at all the words in a sentence simultaneously.

The BERT model undergoes a two-step process:

Pre-Training on Large Data

BERT is pre-trained on large amount of unlabeled text data. The model learns contextual embeddings, which are the representations of words that take into account their surrounding context in a sentence.

BERT engages in various unsupervised pre-training tasks. For instance, it might learn to predict missing words in a sentence (Masked Language Model or MLM task), understand the relationship between two sentences, or predict the next sentence in a pair.

Fine-Tuning on Labeled Data

After the pre-training phase, the BERT model, armed with its contextual embeddings, is then fine-tuned for specific natural language processing (NLP) tasks. This step tailors the model to more targeted applications by adapting its general language understanding to the nuances of the particular task.

BERT is fine-tuned using labeled data specific to the downstream tasks of interest. These tasks could include sentiment analysis, question-answering, named entity recognition, or any other NLP application. The model's parameters are adjusted to optimize its performance for the particular requirements of the task at hand.

How BERT work?

BERT is designed to generate a language model so, only the encoder mechanism is used. Sequence of tokens are fed to the Transformer encoder. These tokens are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors, each corresponding to an input token, providing contextualized representations.

When training language models, defining a prediction goal is a challenge. Many models predict the next word in a sequence, which is a directional approach and may limit context learning. BERT addresses this challenge with two innovative training strategies:

1. Masked Language Model (MLM)
2. Next Sentence Prediction (NSP)

1. Masked Language Model (MLM)

In BERT’s pre-training process, a portion of words in each input sequence is masked and the model is trained to predict the original values of these masked words based on the context provided by the surrounding words.

In simple terms,

Masking words: Before BERT learns from sentences, it hides some words (about 15%) and replaces them with a special symbol, like [MASK].

Guessing Hidden Words: BERT’s job is to figure out what these hidden words are by looking at the words around them. It’s like a game of guessing where some words are missing, and BERT tries to fill in the blanks.

How BERT learns:

BERT adds a special layer on top of its learning system to make these guesses. It then checks how close its guesses are to the actual hidden words.

It does this by converting its guesses into probabilities, saying, “I think this word is X, and I’m this much sure about it.”

Special Attention to Hidden Words

BERT’s main focus during training is on getting these hidden words right. It cares less about predicting the words that are not hidden.

This is because the real challenge is figuring out the missing parts, and this strategy helps BERT become really good at understanding the meaning and context of words.

In technical terms,

1. BERT adds a classification layer on top of the output from the encoder. This layer is crucial for predicting the masked words.
2. The output vectors from the classification layer are multiplied by the embedding matrix, transforming them into the vocabulary dimension. This step helps align the predicted representations with the vocabulary space.
3. The probability of each word in the vocabulary is calculated using the SoftMax activation function. This step generates a probability distribution over the entire vocabulary for each masked position.
4. The loss function used during training considers only the prediction of the masked values. The model is penalized for the deviation between its predictions and the actual values of the masked words.
5. The model converges slower than directional models. This is because, during training, BERT is only concerned with predicting the masked values, ignoring the prediction of the non-masked words. The increased context awareness achieved through this strategy compensates for the slower convergence.

2. Next Sentence Prediction (NSP)

BERT predicts if the second sentence is connected to the first. This is done by transforming the output of the [CLS] token into a 2×1 shaped vector using a classification layer, and then calculating the probability of whether the second sentence follows the first using SoftMax.

a. In the training process, BERT learns to understand the relationship between pairs of sentences, predicting if the second sentence follows the first in the original document.

b. 50% of the input pairs have the second sentence as the subsequent sentence in the original document, and the other 50% have a randomly chosen sentence.

c. To help the model distinguish between connected and disconnected sentence pairs. The input is processed before entering the model:

- A [CLS] token is inserted at the beginning of the first sentence, and a [SEP] token is added at the end of each sentence.
- A sentence embedding indicating Sentence A or Sentence B is added to each token.
- A positional embedding indicates the position of each token in the sequence.

d. BERT predicts if the second sentence is connected to the first. This is done by transforming the output of the [CLS] token into a 2×1 shaped vector using a classification layer, and then calculating the probability of whether the second sentence follows the first using SoftMax.

During the training of BERT model, the Masked LM and Next Sentence Prediction are trained together. The model aims to minimize the combined loss function of the Masked LM and Next Sentence Prediction, leading to a robust language model with enhanced capabilities in understanding context within sentences and relationships between sentences.

How to use BERT model in NLP?

BERT can be used for various natural language processing (NLP) tasks such as:

1. Classification Task

BERT can be used for classification task like sentiment analysis, the goal is to classify the text into different categories (positive/ negative/ neutral), BERT can be employed by adding a classification layer on the top of the Transformer output for the [CLS] token.

The [CLS] token represents the aggregated information from the entire input sequence. This pooled representation can then be used as input for a classification layer to make predictions for the specific task.

2. Question Answering

In question answering tasks, where the model is required to locate and mark the answer within a given text sequence, BERT can be trained for this purpose.

BERT is trained for question answering by learning two additional vectors that mark the beginning and end of the answer. During training, the model is provided with questions and corresponding passages, and it learns to predict the start and end positions of the answer within the passage.

3. Named Entity Recognition (NER)

BERT can be utilized for NER, where the goal is to identify and classify entities (e.g., Person, Organization, Date) in a text sequence.

A BERT-based NER model is trained by taking the output vector of each token form the Transformer and feeding it into a classification layer. The layer predicts the named entity label for each token, indicating the type of entity it represents