Module-4

Semantic Analysis

Introduction

- Semantic Analysis is a subfield of Natural Language Processing (NLP) that attempts to understand the meaning of Natural Language.
- Understanding Natural Language might seem a straightforward process to us as humans.
- However, due to the vast complexity and subjectivity involved in human language, interpreting it is quite a complicated task for machines.
- Semantic Analysis of Natural Language captures the meaning of the given text while taking into account context, logical structuring of sentences and grammar roles.
- Lexical semantics is concerned with the systematic meaning related connections among lexical items, and the internal meaning-related structure of individual lexical items.
- To identify the semantics of lexical items, we need to focus on the notion of lexeme, an individual entry in the lexicon.
- Lexeme should be thought of as a pairing of a particular orthographic and phonological form with some sort of symbolic meaning representation.
- Orthographic form, and phonological form refer to the appropriate form part of a lexeme
- Sense refers to a lexeme’s meaning counterpart.

Relations between word meanings

1. Homonymy is defined as a relation that holds between words that have the same form with unrelated meanings.

Examples:Bat (wooden stick-like thing) vs Bat (flying mammal thing)

Bank (financial institution) vs Bank (riverside)

2. Homophones are the words with the same pronunciation but different spellings.

Examples:write vs right

piece vs peace

3. Homographs are the lexemes with the same orthographic form but different meaning. Ex: bass

4.Polysemy:Multiple related meanings within a single lexeme.

Example: The bank was constructed in 1875 out of local red brick.

I withdrew the money from the bank.

5.Synonym:Two lexemes are synonyms if they can be successfully substituted for each other in all situations.Ex:couch / sofa, big / large

6.Antonym:Senses that are opposites with respect to one feature of their meaning.

Ex:dark / light, short / long

7.Hyponymy:One sense is a hyponym of another if the first sense is more specific, denoting a subclass of the other

Car is a hyponym of vehicle

The dog is a hyponym of animal

mango is a hyponym of fruit

8. Hypernymy

Conversely

vehicle is a hypernym/superordinate of car

animal is a hypernym of dog

fruit is a hypernym of mango

9. Meronymy: an asymmetric, transitive relation between senses

X is a meronym of Y if it denotes a part of Y.

The inverse relation is holonymy.

meronym	holonym
porch	house
wheel	car
leg	chair
nose	face

WordNet

- https://wordnet.princeton.edu/wordnet/
- A hierarchically organized lexical database
- A machine-readable thesaurus, and aspects of a dictionary
- Versions for other languages are under development
- part of speech -no. synsets

➢Noun-82,115

➢Verb-13,767

➢Adjective-18,156

➢Adverb-3,621

Synsets in WordNet

- A synset is a set of synonyms representing a sense
- Example: chump as a noun to mean ‘a person who is gullible and easy to take advantage of’

{chump[1], fool[2], gull[1], mark[9], patsy[1], fall guy[1], sucker[1], soft touch[1], mug[2]}

- Each of these senses share this same gloss.
- For WordNet, the meaning of this sense of chump is this list.

BabbleNet

- BabelNet is an innovative multilingual encyclopedic dictionary, with wide lexicographic and encyclopedic coverage of terms, and a semantic network/ontology which connects concepts and named entities in a very large network of semantic relations, made up of about 22 million entries.
- Conceived within the Sapienza NLP Group, engineered and maintained by Babelscape, BabelNet follows the WordNet model based on the notion of synset (for synonym set), but extends it to contain multilingual lexicalizations.
- Each BabelNet synset represents a given meaning and contains all the synonyms which express that meaning in a range of different languages.

In order to understand the meaning of a sentence, the following are the major processes involved in Semantic Analysis:

1. Word Sense Disambiguation

- In Natural Language, the meaning of a word may vary as per its usage in sentences and the context of the text. Word Sense Disambiguation involves interpreting the meaning of a word based upon the context of its occurrence in a text.
- For example, the word ‘Bark’ may mean ‘the sound made by a dog’ or ‘the outermost layer of a tree.’Likewise, the word ‘rock’ may mean ‘a stone‘ or ‘a genre of music‘ – hence, the accurate meaning of the word is highly dependent upon its context and usage in the text.
- Thus, the ability of a machine to overcome the ambiguity involved in identifying the meaning of a word based on its usage and context is called Word Sense Disambiguation.

2. Relationship Extraction: It involves firstly identifying various entities present in the sentence and then extracting the relationships between those entities

- Sense ambiguity
- Many words have several meanings or senses

Example: The meaning of bass depends on the context

- Are we talking about music, or fish?
- An electric guitar and bass player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps.And it all started when fishermen decided the striped bass in Lake Mead were too skinny.
- Disambiguation
- The task of disambiguation is to determine which of the senses of an ambiguous word is invoked in a particular use of the word.
- This is done by looking at the context of the word's use.

Algorithms to Word Sense Disambiguation

- Knowledge Based Approaches
  - Overlap Based Approaches
  - Lesk's algorithm
- Machine Learning Based Approaches
  - Supervised Approaches(Naïve Bayes, Decision List)
  - Semi-supervised Algorithms(Yarowsky)
  - Unsupervised Algorithms(Hyperlex)

- Overlap Based Approaches

➢Require a Machine Readable Dictionary (MRD).

➢Find the overlap between the features of different senses of an ambiguous word (sense bag) and the features of the words in its context(context bag).

➢The features could be sense definitions, example sentences, hypernyms etc.

➢The features could also be given weights.

➢The sense which has the maximum overlap is selected as the contextually appropriate sense.

- The Lesk algorithm follows a simple yet effective approach to disambiguate words. Let's break down the steps involved:

1. Step 1: Preprocessing
    1. Tokenize the input text into individual words.
    2. Remove stop words and punctuation marks to focus on content words.
2. Step 2: Retrieve Definitions
    1. Retrieve dictionary definitions for the ambiguous word from a lexical resource like WordNet.
    2. WordNet is a large lexical database that organizes words into synsets (sets of synonyms).
3. Step 3: Calculate Overlaps
    1. Compare the definitions of the ambiguous word with the definitions of its context words.
    2. Calculate the overlap between the glosses (definitions) of the ambiguous word and the glosses of its context words.
    3. The overlap can be calculated using various techniques, such as counting common words or using semantic similarity measures.
4. Step 4: Select the Best Sense
    1. Choose the sense with the highest overlap as the disambiguated sense for the ambiguous word.
    2. The sense with the highest overlap indicates the most relevant meaning in the given context

- **Advantages and Limitations of the Lesk Algorithm**
- The Lesk algorithm offers several advantages, making it a popular choice for word disambiguation:
- Simplicity: The algorithm is relatively simple to implement and understand, making it accessible to both beginners and experts.
- No Supervised Training: Unlike some other disambiguation techniques, the Lesk algorithm does not require labeled training data.
- Contextual Understanding: By considering the context, the algorithm captures the nuances of word meaning based on the surrounding words.

However, the Lesk algorithm also has certain limitations:

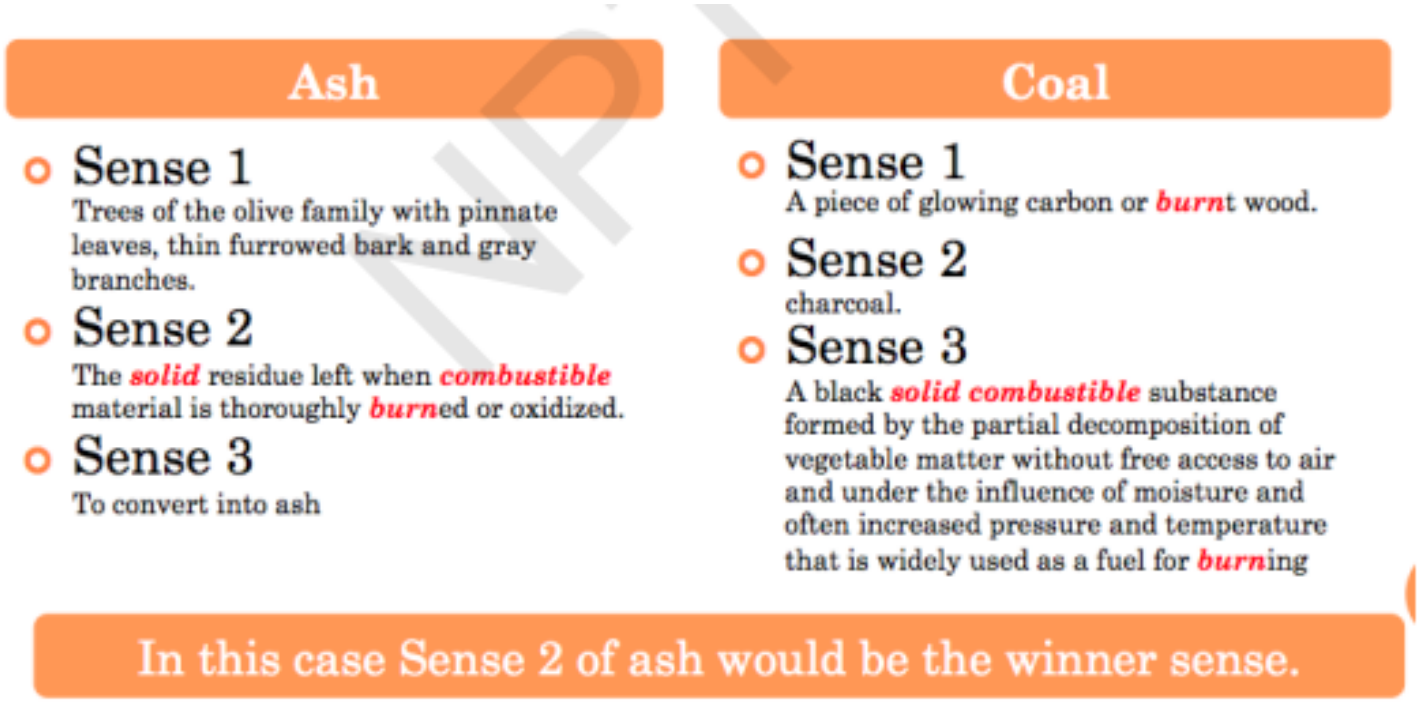
- Reliance on Dictionary Definitions: The algorithm relies heavily on the availability and accuracy of dictionary definitions. Inaccurate or incomplete definitions can affect the disambiguation accuracy.
- Lack of Polysemy Handling: The algorithm may struggle with words that have multiple meanings, especially when the context is not sufficient to disambiguate them

- **Applications of the Lesk Algorithm**
- The Lesk algorithm finds applications in various NLP tasks, including:

1. **Machine Translation**: Accurate word disambiguation improves the translation quality by selecting the appropriate meaning of words in the source language.
2. **Information Retrieval**: Disambiguating query terms helps retrieve more relevant documents by matching the correct sense of words in the search process.
3. **Text Summarization**: Resolving word ambiguity aids in generating concise and accurate summaries by capturing the intended meaning

Lesk's Algorithm

- Sense Bag: contains the words in the definition of a candidate sense of the ambiguous word.
- Context Bag: contains the words in the definition of each sense of each context word.
- Example:On burning coal we get ash.



Supervised: The assumption behind supervised approaches is that the context can supply enough evidence to disambiguate words on its own.

Supervised Word Sense Disambiguation (WSD) methods 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 assign a sense to a target word based on the context in which it appears.

Neural Network: Neural networks such as feedforward, recurrent neural, 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.

Naive Bayes is a probabilistic machine learning algorithm that can be used to help resolve Word Sense Disambiguation (WSD) problems. WSD is the task of determining the correct meaning of a word in context when the word has multiple possible meanings or senses. Here's how Naive Bayes can assist in WSD:

Feature Representation: To apply Naive Bayes for WSD, you need to represent the words in the context (surrounding words) of the ambiguous word as features. These features should capture the information that can discriminate between different senses of the word

Training: You need a labeled dataset where each instance contains the context of an ambiguous word and its correct sense label. Naive Bayes learns the conditional probabilities of features given each sense. It calculates how likely each feature is to occur given a particular sense of the word

Classification: When you encounter a new instance where the sense of the ambiguous word needs to be disambiguated, Naive Bayes calculates the probability of each sense given the observed features in the context. It uses Bayes' theorem to compute these probabilities.

Decision: The sense with the highest probability becomes the predicted sense for the ambiguous word.

- **Key advantages of using Naive Bayes for WSD**:
- - **Simplicity**: Naive Bayes is relatively simple and easy to implement.
- - **Efficiency**: It can handle high-dimensional feature spaces efficiently.
- - **Independence Assumption**: Despite its "naive" assumption of feature independence, it often works surprisingly well in practice, especially when a large amount of data is available.
- However, it's important to note that Naive Bayes may not capture complex dependencies between features or senses, which can limit its performance on highly nuanced WSD tasks. In such cases, more advanced machine learning models, like neural networks and deep learning approaches, may be more suitable.

- **Support Vector Machines (SVMs)** can be used to help resolve Word Sense Disambiguation (WSD) problems by classifying the correct sense of a word within a given context. Here's how SVMs can assist in WSD:

- 1. **Feature Extraction**: SVMs require input data in the form of feature vectors. In the context of WSD, words and their surrounding context are represented as features. These features can include surrounding words, their parts of speech, word embeddings, or other relevant linguistic information.

- 2. **Training Data**: To use SVMs for WSD, you need a labeled dataset where each word is associated with its correct sense in various contexts. This dataset is used for training the SVM

- 3. **Binary Classification**: SVM is a binary classification algorithm, meaning it's designed to separate data into two classes. In the context of WSD, you'll typically treat each sense of a word as a separate class. For example, if a word has three senses, you'd create three binary classifiers, each distinguishing one sense from the others.

- 4. **Training SVMs**: For each sense of a word, you train a separate SVM classifier using the feature vectors and labeled data. The SVM learns to find a hyperplane that best separates the data points representing the different senses.

- 5. **Prediction**: When you encounter an ambiguous word in a new context, you extract the relevant features for that context. Then, you feed these features into each of the trained SVM classifiers. The classifier that produces the highest confidence score or margin is considered the predicted sense for that context.

- 6**. Evaluation**: The performance of your WSD system using SVMs is typically evaluated using standard metrics like accuracy, precision, recall, and F1-score on a separate test dataset.

- SVMs have been used successfully in WSD tasks because they are effective at finding decision boundaries in high-dimensional feature spaces, which can capture the subtle differences in word senses. However, more recent machine learning approaches like deep learning methods, such as recurrent neural networks (RNNs) and transformers, have also been employed and often outperform traditional SVM-based approaches in WSD tasks, thanks to their ability to capture complex contextual relationships.

- **The Decision Tree algorithm** can be used to help resolve Word Sense Disambiguation (WSD) by creating a hierarchical structure that classifies words based on their context and meaning. Here's how it can be applied:

- 1. **Data Preparation**: Collect a dataset that includes words with multiple senses and their context. Each instance in the dataset should be labeled with the correct sense of the word in that context.

- 2. **Feature Extraction**: Extract features from the context of the word. These features can include the surrounding words, part-of-speech tags, syntactic structures, and more.

- 3. **Training**: Use the dataset to train a Decision Tree classifier. The Decision Tree will learn to make decisions based on the extracted features to classify words into their correct senses.
- 4. **Decision Making**: During the disambiguation process, when you encounter an ambiguous word in a sentence, you can use the trained Decision Tree to make a decision about its sense. You input the context features of the word into the Decision Tree, and it follows the learned rules to classify the word into one of its senses.

- 5. **Evaluation**: You can evaluate the performance of your WSD system by comparing its classifications to the ground truth labels in your dataset.

- 6. **Fine-Tuning**: Depending on the performance, you can fine-tune the Decision Tree model by adjusting hyperparameters or trying other tree-based algorithms like Random Forests or Gradient Boosting Trees.

- The Decision Tree algorithm helps resolve WSD by creating a decision-making process based on context features, enabling it to make sense distinctions for ambiguous words. However, it's important to note that while Decision Trees can be effective for certain WSD tasks, more advanced machine learning techniques like deep learning with neural networks have shown better performance in recent years for complex WSD tasks.

- **The Yarowsky algorithm** is a supervised machine learning approach used to resolve Word Sense Disambiguation (WSD) problems. WSD is the task of determining the correct sense or meaning of a word in context, especially when a word has multiple possible meanings.
- 1. **Initial Seed Set**: It starts with a small labeled dataset where word senses are manually annotated. This dataset serves as the initial seed set of labeled examples.
- 2. **Feature Extraction**: Features are extracted from the context of the ambiguous word. These features can include surrounding words, part-of-speech tags, and other linguistic information.
- 3. **Training**: The algorithm uses the initial seed set and features to train a classifier, often a decision tree or a Naive Bayes classifier. The classifier learns patterns that associate the context features with specific word senses.
- 4. **Bootstrapping**: The trained classifier is then used to predict word senses for a larger, unlabeled dataset. The predictions that the algorithm is confident about are added to the labeled dataset, expanding it.
- 5. **Iteration**: Steps 3 and 4 are repeated for multiple iterations, gradually increasing the size of the labeled dataset.
- 6. **Convergence**: The process continues until it reaches a predefined stopping criterion, such as a certain number of iterations or when performance on a validation set plateaus.
- 7. **Final Classifier**: The final classifier is then used to disambiguate word senses in new, unseen contexts. The Yarowsky algorithm leverages the iterative nature of bootstrapping to improve its accuracy over time. It's effective in WSD because it learns from a limited set of labeled data and leverages the context in which words appear to make sense distinctions. However, it does require a reasonably large amount of unlabeled data for bootstrapping to be effective.

Naïve Bayes for WSD

- A Naïve Bayes classifier chooses the most likely sense for a word given the features of the context:

$$\hat{s} = \arg\max_{s \in S} P(s|f)$$

- Using Bayes' law, this can be expressed as:

$$\hat{s} = \arg\max_{s \in S} \frac{P(s)P(f|s)}{P(f)}$$

$$= \arg\max_{s \in S} P(s)P(f|s)$$

- The 'Naïve' assumption: all the features are conditionally independent, given the sense':

$$\hat{s} = \arg\max_{s \in S} P(s) \prod_{j=1}^{n} P(f_j|s)$$

Training for Naïve Bayes

- 'f' is a feature vector consisting of:
  - POS of w
  - Semantic and Syntactic features of w
  - Collocation vector (set of words around it) → next word (+1), +2, -1, -2 and their POS's
  - Co-occurrence vector
- Set parameters of Naïve Bayes using maximum likelihood estimation (MLE) from training data

$$P(s_i) = \frac{count(s_i, w_j)}{count(w_j)}$$

$$P(f_j|s_i) = \frac{count(f_j, s_i)}{count(s_i)}$$

Decision List Algorithm

- Based on 'One sense per collocation' property
  - Nearby words provide strong and consistent clues as to the sense of a target word
- Collect a large set of collocations for the ambiguous word
- Calculate word-sense probability distributions for all such collocations
- Calculate the log-likelihood ratio

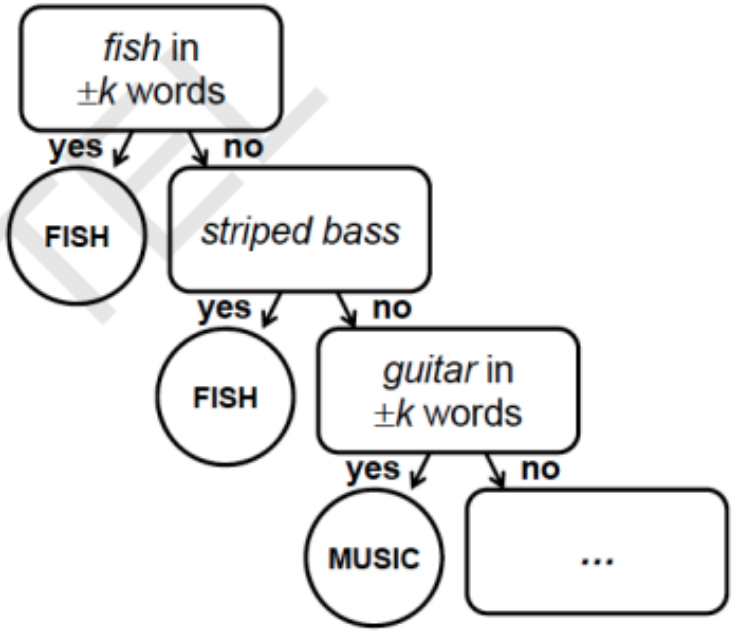
$$log(\frac{P(Sense-A|Collocation_i)}{P(Sense-B|Collocation_i)})$$

- Higher log-likelihood ⇒ more predictive evidence
- Collocations are ordered in a decision list, with most predictive collocations ranked highest

Example

Example: discriminating between bass (fish) and bass (music):

Context	Sense
fish in ±*k* words	FISH
striped bass	FISH
guitar in ±*k* words	MUSIC
bass player	MUSIC
piano in ±*k* words	MUSIC
sea bass	FISH
play bass	MUSIC
river in ±*k* words	FISH
on bass	MUSIC
bass are	FISH



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.

Minimally Supervised WSD - Yarowsky

- Annotations are expensive!
- “Bootstrapping” or co-training

➢ Start with a (small) seed, learn the decision list

➢ Use a decision list to label the rest of the corpus

➢ Retain ‘confident’ labels, and treat them as annotated data to learn new decision list

➢ Repeat ...

- Heuristics (derived from observation):

➢ One sense per discourse(A word tends to preserve its meaning across all its occurrences in a given discourse)

➢ One sense per collocation(A word tends to preserve its meaning when used in the same collocation Strong for adjacent collocations and Weaker as the distance between the words increases)

Example

- Disambiguating plant (industrial sense) vs. plant (living thing sense)
- Think of seed features for each sense
  - Industrial sense: co-occurring with ‘manufacturing’
  - Living thing sense: co-occurring with ‘life’
- Use ‘one sense per collocation’ to build initial decision list classifier
- Treat results (having high probability) as annotated data, train new decision list classifier, iterate

HyperLex

- Key Idea: Word Sense Induction
- Instead of using “dictionary defined senses”, extract the “senses from the corpus” itself
- These “corpus senses” or “uses” correspond to clusters of similar contexts for a word.

Detecting Root Hubs

- Different uses of a target word form highly interconnected bundles (or high density components)
- In each high density component one of the nodes (hub) has a higher degree than the others.
- Step 1: Construct co-occurrence graph, G.
- Step 2: Arrange nodes in G in decreasing order of degree.
- Step 3: Select the node from G which has the highest degree. This node will be the hub of the first high density component.
- Step 4: Delete this hub and all its neighbors from G.
- Step 5: Repeat Step 3 and 4 to detect the hubs of other high density components