Yarowsky Algorithm and Naive Bayes Classifier (WSD),Hyperlex

By Prof Nirali Arora

Introduction

David Yarowsky algorithm for word sense disambiguation (WSD) is a supervised learning approach that primarily focuses on leveraging context to determine the meaning of words with multiple senses.

It falls under the category of semi supervised learning .

Key principles

Yarowsky algorithm is based on two key principles :

One sense per collocation: This principle states that a given word tends to have a consistent sense when it appears in the same collocation(specific lexical or syntactic context)

One sense per discourse : This principle states that a given word tends to maintain the same sense throughout a particular discourse or document.

Key concepts

Key Concepts:

1. **Word Senses: Each word may have multiple meanings (e.g., "bank" as a financial institution vs. "bank" as the side of a river).**
2. **Contextual Features: The surrounding words (context) help identify which sense of a word is intended**

Algorithm overview

Training Data: The algorithm requires a labeled dataset where each instance includes a target word and its corresponding sense.

Context Window: For each instance, a window of words around the target word is considered. This context is crucial for determining the sense.

Feature Extraction: The algorithm extracts features from the context window. This might include:

- Surrounding words
- Part of speech tags
- Syntactic structures

Yarowsky algorithm continued

Supervised Learning: Using the labeled training data, Yarowsky applies a machine learning classifier (like decision trees or Naive Bayes) to learn the relationship between the extracted features and the word senses.

Unlabeled Data Expansion: A notable aspect of Yarowsky's method is its ability to leverage both labeled and unlabeled data. It employs a semi-supervised approach, where the classifier can also be applied to unlabeled data, iteratively refining its understanding of word senses.

Yarowsky Algorithm continued

Iterative Disambiguation: The algorithm can iteratively disambiguate words, improving accuracy with each pass through the data, as it identifies new instances of word senses based on the learned features.

Finally, the output consists of the disambiguated senses for the target words across the dataset, ready for further processing or analysis.

Workflow

Initial Training: You start with 1000 labeled examples of "bank," then extract context features and train a classifier.

Disambiguate Unlabeled Instances: You have a corpus with 10,000 occurrences of "bank" without labels. Apply the classifier to predict senses based on context.

Refine with Self-Training: Take the confident predictions (say 2000 instances), add them to the training data, and retrain.

Repeat: Continue this process until convergence or until no further improvement is noted.

Example of yarowsky algorithm

Step 1: Data Preparation

Labeled Data: Suppose you have the following labeled sentences:

1. "I went to the **bank** to deposit some money." (Sense: financial institution)
2. "The river **bank** was eroded by the flood." (Sense: side of a river)

Unlabeled Data: You also have the following sentences, where "bank" is used but not labeled:

1. "The fisherman sat on the **bank**."
2. "She transferred money to her savings **bank**."
3. "The **bank** of the river was beautiful in spring."

Feature extraction

Step 2: Feature Extraction

For each occurrence of "bank" in the context sentences, extract features from a context window of, say, two words to the left and right.

- **For "bank" in the first unlabeled sentence:**
  - **Context: ["fisherman", "sat", "on", "the"]**
  - **Features might include:**
    - **Surrounding words: "fisherman", "sat", "on", "the"**
    - **POS tags: Noun (bank), Verb (sat)**
- **For "bank" in the second unlabeled sentence:**
  - **Context: ["transferred", "money", "to", "her"]**
  - **Features might include:**
    - **Surrounding words: "transferred", "money", "to", "her"**
    - **POS tags: Verb (transferred), Noun (money)**

Yarowsky algorithm example

Initial Classification

Using the labeled data, you train a classifier (e.g., Naive Bayes) based on the features extracted. The classifier learns to associate the context features with the two senses of "bank."

Step 4: Disambiguate Unlabeled Data

Now, apply the trained classifier to the unlabeled instances:

1. **For "The fisherman sat on the bank":**
   - **Features are extracted and fed into the classifier.**
   - **The classifier predicts this instance as "side of a river" based on similar features from the training data.**
2. **For "She transferred money to her savings bank":**
   - **The classifier predicts "financial institution."**
3. **For "The bank of the river was beautiful in spring":**
   - **The classifier predicts "side of a river."**

Iterative refinement

Iterative Refinement

After labeling the unlabeled data:

- **Assume the classifier confidently predicts:**
  - **"bank" in the fisherman sentence as riverbank.**
  - **"bank" in the savings sentence as financial institution.**

These predictions can be added back to the training data to create a more extensive labeled dataset.

Step 6: Self-Training

Retrain the classifier with this expanded dataset and repeat the process of disambiguating any new unlabeled instances. Over multiple iterations, the classifier should improve in accuracy as it learns from the newly labeled examples.

In this example, Yarowsky algorithm uses a small amount of labeled data to bootstrap a larger disambiguation process. By leveraging the context of words and applying iterative learning, it can effectively determine the correct sense of "bank" in various contexts. This approach is particularly useful when labeled data is limited, as it enhances the classifier’s ability to generalize and improve over time.

Naive Bayes Classifier for (WSD)

The Naive Bayes classifier is a simple yet effective probabilistic model often used for tasks like word sense disambiguation (WSD).

What is Naive Bayes?

1. **Probabilistic Model: Naive Bayes is based on Bayes' Theorem, which helps calculate the probability of a particular outcome given prior knowledge. In WSD, the outcome is the correct sense of a word.**
2. **Naive Assumption: The "naive" part refers to the assumption that all features (in this case, words in the context of the target word) are independent of each other. While this assumption isn't strictly true in natural language, it simplifies calculations and often works well in practice.**

Naive Bayes algorithm steps

Step 1: Data Preparation

- **Labeled Data: You need a dataset where the target word is used in different contexts and labeled with its correct senses. For example, "bank" could be labeled as "financial institution" or "riverbank."**

Step 2: Feature Extraction

- **Context Features: For each occurrence of the target word, extract features based on the surrounding words (context). This could include:**
  - **The words immediately before and after the target word.**
  - **Parts of speech of those words.**

Step 3: Training the Classifier

1. **Calculate Probabilities:**
   - **For each sense of the word, calculate the prior probability (how often each sense occurs) and the likelihood of each feature given that sense. For example:**
     - **P(Sense): The probability of each sense (how frequently each sense appears in the training data).**
     - **P(Feature|Sense): The probability of seeing each feature in the context when the word has a particular sense.**

Use Bayes' Theorem:

- **For a new instance, apply Bayes' theorem to compute the probability of each sense given the features: $P(Sense \mid Features) \propto P(Sense) \times P(Features \mid Sense)$**

Step 4: Making Predictions

- **For a new occurrence of the target word, extract its context features and calculate the probabilities for each sense using the trained model.**
- **Choose the sense with the highest probability as the predicted sense for that instance.**

Example

Let’s say you have the word "bank" in two different sentences:

1. **"The river bank was eroded."**
2. **"I went to the bank to withdraw money."**

Training Data:

- **Sense 1 (river bank): Occurs in contexts with features like “river,” “water.”**
- **Sense 2 (financial institution): Occurs with features like “money,” “withdraw.”**

Feature Extraction:

- **For "The river bank was eroded," you extract features: ["river", "was", "eroded"].**
- **For "I went to the bank to withdraw money," you extract: ["went", "to", "withdraw", "money"].**

Calculate probabilities based on labeled examples from your training data.

For a new instance: "The fisherman sat by the bank," extract features and compute probabilities for both senses of "bank."

If the features lead to a higher probability for "river bank," that’s the predicted sense.

Conclusion

The Naive Bayes classifier is a powerful tool for word sense disambiguation. It uses the independence assumption to simplify the calculation of probabilities, allowing it to effectively predict the correct meaning of words based on their context. Despite its simplicity, it often performs well in practice, especially when there is a sufficient amount of training data.

Hyperlex

The Hyperlex algorithm is a method designed for word sense disambiguation (WSD) that leverages lexical relations within a semantic network, particularly focusing on the relationships among words in a hierarchical structure like WordNet.

It is an unsupervised wsd algorithm proposed by Peter Vosen in 2002

It clusters the word contexts to discover different senses of the word

Hyperlex algorithm overview

Lexical Semantics: Hyperlex operates based on the idea that words are connected through various semantic relationships (e.g., synonyms, antonyms, hypernyms, and hyponyms) in a semantic network.

Contextual Clustering: The algorithm uses the context in which a word appears to create clusters of senses, helping to identify the most relevant sense for the given context.

Hyperlex algorithm working

Input: The algorithm takes a target word and its surrounding context from a sentence.

Semantic Relations: It looks up the target word in a semantic network (like WordNet) to retrieve its possible senses along with their relationships to other words.

Hyperlex algorithm continued

Context Analysis:

- **The algorithm extracts features from the context surrounding the target word (like nearby words).**
- **It identifies other words in the context and checks their relationships to the possible senses of the target word.**

Hyperlex algorithm

Similarity Measurement: Hyperlex computes similarity scores between the context words and the senses of the target word based on their semantic relationships. This often involves:

- **Calculating distance in the semantic network.**
- **Using metrics to quantify how closely related the context words are to each possible sense.**

Clustering: The algorithm groups senses based on their relatedness to the context. The sense that has the highest similarity to the context words will likely be the one chosen.

Disambiguation: Finally, Hyperlex selects the most relevant sense for the target word based on the analysis, improving the accuracy of the disambiguation process

Advantages of Hyperlex

- **Rich Semantic Information: By utilizing a semantic network, Hyperlex can effectively capture the nuances of word meanings based on their relationships with other words.**
- **Robustness: It can handle cases where words have multiple meanings and provide a more informed disambiguation compared to simpler methods.**

Applications

Hyperlex is particularly useful in tasks like:

- **Natural language processing (NLP)**
- **Information retrieval**
- **Machine translation**