CS 5322 Program 3 Report

Harley Gribble

Introduction

Word Sense Disambiguation (WSD) is the task of determining which sense (meaning) of a word is used in a given context. It is a long-standing and fundamental problem in natural language processing (NLP) that lies at the core of many language understanding applications such as machine translation, information retrieval, and question answering. For example, the word "deed" can refer to a legal document or to a noble action, and selecting the wrong interpretation can severely distort downstream tasks.

WSD is challenging for several reasons:

- **Polysemy and ambiguity**: Many words have multiple valid senses that are semantically distinct but grammatically identical, making them hard to separate without deep context.
- **Sparse training data**: Manually labeled examples are expensive to collect, and many word senses are underrepresented in corpora.
- **Subtle distinctions**: Some word senses differ in very fine-grained ways, often requiring world knowledge or pragmatic inference to resolve.

Despite its difficulty, WSD remains an active research topic because accurate sense prediction improves language understanding across many domains. This project explores a supervised learning approach to WSD, using both neural and symbolic models to classify senses for three highly polysemous target words: **camper**, **conviction**, and **deed**.

Overview

This project implements a supervised word sense disambiguation (WSD) system for three target words: **camper**, **conviction**, and **deed**. For each word, two senses are defined and

example-labeled training data was created (100 examples per sense). The goal is to build models that predict the correct sense (1 or 2) for a new sentence containing the target word.

Approach

Data Preparation

- Manually curated 100 example sentences for each sense of each word using realistic, diverse context.
- Paraphrased examples and introduced lexical and syntactic variation to ensure generalization.
- Each dataset is stored as a plain text file in the format:

```
word (POS)
1: definition for sense 1
2: definition for sense 2
1 sentence using sense 1
...
2 sentence using sense 2
...
```

Preprocessing & Feature Engineering

For `camper` and `conviction`:

- Used **Sentence-BERT embeddings** (`all-MiniLM-L6-v2`) to convert each sentence into a dense vector.
- SBERT provides rich contextual information ideal for detecting subtle meaning differences.
- Minimal preprocessing required, just whitespace trimming and batch encoding.

For `deed`:

- A hybrid pipeline was used to capture both symbolic and contextual information:
  - `LemmaTransformer`: custom scikit-learn transformer for word normalization.
  - `GlossSimilarity`: feature comparing sentence tokens with WordNet glosses for both senses.
  - Combined features using `FeatureUnion` to feed into the classifier.

This symbolic addition was necessary because SBERT embeddings alone underperformed on highly ambiguous inputs for "deed".

Classification Algorithms

- **Camper & Conviction**: Used a **Multi-Layer Perceptron (MLPClassifier)** from scikit-learn on SBERT vectors. MLPs are effective for low-dimensional embeddings and capture nonlinear boundaries well.
- **Deed**: Used a **stacked ensemble** with:
  - Logistic Regression (interpretable, baseline)
  - Random Forest (nonlinear, robust to feature interactions)
- Classifiers were validated during prototyping using cross-validation; final models were trained on all 200 examples per word.

Training

Training Script Overview (`train.py`)

This script handles training and saving word sense disambiguation (WSD) models for three target words: **camper**, **conviction**, and **deed**. Each section is modular and optimized for the word's disambiguation challenge.

Preprocessing Components

- `LemmaTransformer`: Lemmatizes each token to normalize word forms.
- `WordNetOverlap`: Computes the number of overlapping words between the sentence and the WordNet gloss of each sense.
- `WindowFeatures`: Extracts local lexical features (e.g., window size and article counts) near the target word.
- `GlossSimilarity`: Encodes glosses and sentences using SBERT, returning cosine similarity scores to each gloss definition.

These are used especially in the symbolic pipeline for the word "deed."

Data Functions

- `load_data(path)`: Loads labeled sentences from a formatted file where each line begins with the correct sense label.
- `synonym_augment(sents, labels, n_aug)`: Generates additional examples by replacing words in the sentence with WordNet synonyms, useful for increasing diversity and training size.

Model Builders

- `build_conviction_model()`:
  - Loads raw labeled data.
  - Encodes with SBERT.
  - Trains an MLP classifier.
  - Saves the model as `conviction_mlp_sbert.joblib`.
- `build_camper_model()`:
  - Loads and augments data via synonym replacement.
  - Encodes with SBERT.
  - Trains a deeper MLP model.
  - Saves as `camper_mlp_sbert_aug.joblib`.
- `build_deed_model()`:
  - Loads and heavily augments training data.
  - Applies a multi-feature symbolic pipeline using TF-IDF, gloss similarity, and WordNet overlap.
  - Trains a random forest classifier.
  - Saves as `deed_stack_pipe.joblib`.

Execution Trigger

The script ends with a `__main__` block that trains all three models when run directly, ensuring reproducibility and modular testing.

WSD Prediction and Disambiguation

Each WSD function (`WSD_Test_camper`, etc.) in `cs5322s25.py`:

- Accepts a list of raw input sentences.

- Loads the relevant pre-trained model (MLP or pipeline) **inside the function** as required.
- Converts the sentence into features (via SBERT or symbolic pipeline).
- Returns a list of predicted integer sense labels (**1** or **2**) in order.

In practical use, disambiguation occurs by transforming each input sentence into a structured feature representation (via SBERT or handcrafted features), and then passing that vector into the trained classifier, which outputs a sense label (1 or 2). This process is repeated for each input sentence, ensuring label order is preserved.

All models are saved as `.joblib` files and loaded dynamically.

Evaluation Results

We tested the system on multiple unseen test sets per word:

- Each set had 10 sentences: 5 for each sense.
- "New" and "newer" sets test generalization.
- "Hard" test sets contain borderline or misleading phrasing.

Word	Accuracy Range	Notes
camper	80%–100%	Strong and stable on varied sentence structures
conviction	60%–100%	Slightly weaker on borderline usage, strong on clear-cut inputs
deed	90%–100%	Gloss-enhanced classifier improved robustness dramatically

Sample Prediction Outcomes

Camper

- *"He installed solar panels on his camper."* → 1 (vehicle)

- *"Each camper must sign in before breakfast."* → 2 (person)
- *"After hours on the trail, the group set up their camper."* → Predicted 2, true 1

Conviction

- *"Her conviction that honesty matters inspired us all."* → 1 (belief)
- *"It was a speech full of conviction and legal terms."* → Predicted 2, true 1

Deed

- *"He signed the deed to the property."* → 1 (document)
- *"Her brave deed saved a child."* → 2 (action)

Final Testing Procedure

The file `cs5322s25.py` includes the function `run_test("Harley", "Gribble")` which:

- Automatically identifies all `<word>_test.txt` files in the working directory.
- Calls the appropriate `WSD_Test_*` function per file.
- Writes output to `result_<word>_harleygribble.txt` with one label per line (1 or 2).

Example Command

```
python cs5322s25.py
```

This will output:

- `result_camper_harleygribble.txt`
- `result_conviction_harleygribble.txt`
- `result_deed_harleygribble.txt`

Summary and Reflection

This project highlights the nuanced tradeoffs between deep contextual modeling and traditional symbolic reasoning in word sense disambiguation. Sentence-BERT (SBERT) proved to be an effective and efficient baseline, offering strong performance with minimal preprocessing and fast training times. Its ability to capture rich semantic context from sentences made it particularly well-suited for distinguishing subtle differences in meaning, especially for words like camper and conviction. However, our experiments also revealed that for certain words—such as deed, where sense distinctions rely more on external knowledge or gloss definitions—purely neural methods

may fall short. In these cases, incorporating symbolic features derived from semantic resources like WordNet, including gloss comparisons and lexical overlap, added crucial interpretability and robustness. Each model was carefully adapted to the particular challenges posed by its target word, demonstrating that no single approach universally dominates and that hybrid strategies can yield more reliable disambiguation in practice.

Future Improvements

- Add confidence scores and fallback heuristics for borderline cases.
- Augment training with real-world corpora or crowd-labeled examples.
- Try ensemble voting across multiple model types (SBERT, TF-IDF, gloss match).