1. **Introduction**

A fundamental characteristic of language is that words can have more than one distinct meaning. The lexical ambiguity is especially obvious to anyone who gets the joke when they hear a pun. The 121 most frequently used English nouns, which account for about 20% occurrences in real text, have on average 7.8 meanings each (1). However, for anyone who can fluently use English, the potential for ambiguous readings tends to be completely unnoticeable. This fact demonstrates that even though words have multiple meanings in principle, there is very little ambiguity to a real person in actual text. Step into the time of information explosion, nowadays people are trying to make machines “smart” enough to do anything for human beings. Making the machines to determine the meaning of every word in context is not a trivial task, computer scientists and software engineers thus came up with different algorithms to realize this goal. This type of problem in computational linguistics filed is called word sense disambiguation (WSD), which is essentially a task of classification. Word senses are classes, the context provides the evidence, and each occurrence of a word is assigned to one or more classes based on the evidence (2). Some common algorithms that currently used to handle WSD problem are including HyperLex algorithm, Lesk algorithm, Yarowsky algorithm, etc. In this project, we will focus on explaining (section 2), implementing (section 3) and evaluating (section 4) Yarowsky algorithm. We will implement Yarowsky algorithm in Spark with both Scala (for CS651) and Python (for CS631) drivers. Dataset for testing and code can be found in GitHub <https://github.com/aixeuy/DI-C-project>, and will be discussed in detail in section 3.

1. **Original Yarowsky algorithm**

The original Yarowsky algorithm is an unsupervised learning algorithm for WSD built by Professor David Yarowsky from University of Pennsylvania. He claims that this algorithm rivals the performance of supervised techniques that require hand annotations when trained on unannotated English text. This algorithm is based on two properties of human language:

1. One sense per collocation: some collocations are excellent features of the context that can be strong predictors for one sense or another.
2. One Sense Per Discourse: ambiguous tokens of the same type tend to be correlated within a document (i.e. discourse)

Moreover, as mentioned above, since a small proportion of English nouns are frequently used and account for about one fifth of occurrences in real text, we can at least conclude that English language is highly redundant. Therefore, the sense of a word is effectively overdetermined by 1) and 2) above. Section 2.1 and 2.2 discuss in detail how the two properties help to build Yarowsky Algorithm, section 2.3 explains the original algorithm step by step.

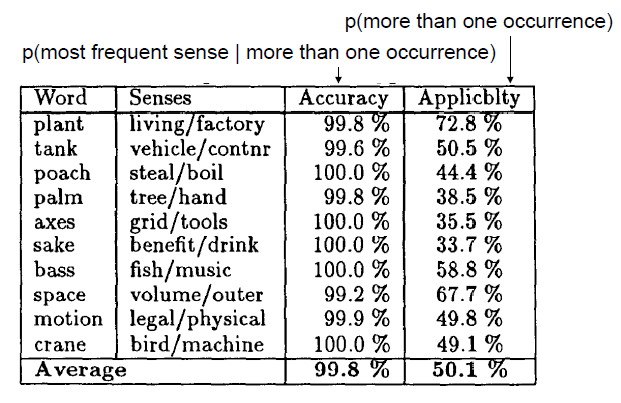
* 1. **One sense per collocation**

Prof. Yarowsky observed and quantified the strong tendency that words will exhibit only one sense in a given collocation in his work in 1993. The distance and adjacency of collocation, the predicate-argument relationship, and the burstiness of the words, all affect the influence of collocation. This property of language is highly reliable and is extremely useful for WSD problems. Based on this property, Prof. Yarowsky has built a supervised algorithm called decision-list algorithm in 1994, and it is used as a component of Yarowsky algorithm.

A decision list is an ordered set of rules with form that if some object has a particular feature f, then predict class label k. The rules can be ordered based on weights assigned. The weights are evaluated using log-likelihood ratio , and we apply the rule with the highest weight. Note that these weights are the parameters of the classifier that to be learned by training algorithm. The Yarowsky algorithm is self-trained with decision list base classifier, and we will discuss in detail in section 2.3.

* 1. **One sense per discourse**

Observed by Gale, Church and Yarowsky, words tend to exhibit only one sense in a given document (or discourse), which means we can take advantage of this regularity in conjunction for each word. This property is weaker than the collocation property mentioned above since it can be overridden when local evidence is strong, thus should not be used as a hard constraint. Prof. Yarowsky tested the one sense per discourse hypothesis on a set of 37,232 examples over a 3-year period for the claim’s accuracy and applicability. Accuracy measures how often a word takes on the majority sense for a discourse when it occurs more than once in this discourse. Applicability measures how often a word occurs more than once in a discourse. The following table demonstrates that the claim of one sense per discourse holds for at least these 10 words with high reliability.



* 1. **Original Yarowsky Algorithm**

In the real world, instead of occurring in one collocation that indicates the sense, a word tend to occur in multiple such collocations. Beginning with a small set of seed examples that representing different meanings of a word, one can expand the seed examples using the two properties of language mentioned above with additional data. Steps of the original Yarowsky algorithm are as following:

Step1:

Identify all occurrences of the given polysemous word from the text corpus and store their context as lines in an untagged training set.

Step2:

Identify (can hand-tag) a relatively small number of training examples to represent each sense of the word. There are several strategies that require minimal or no human participation in identifying seeds, including 1) use a single defining collocate for each class, 2) use words in dictionary definitions and 3) label salient corpus collocations. We use these training examples as seeds and call it the initial state (see Figure 1).

Setp3:

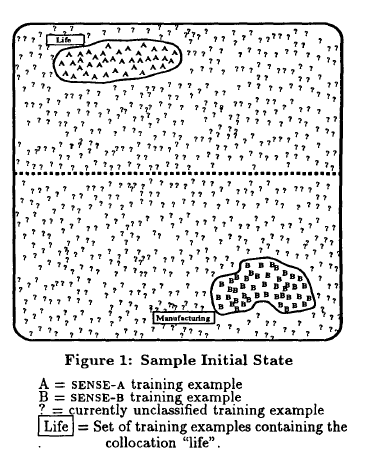
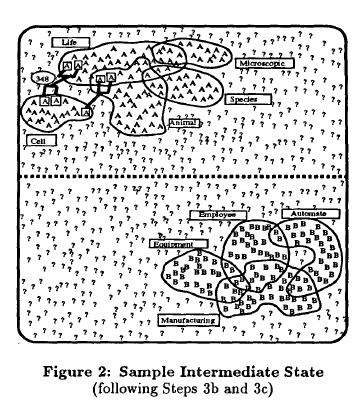
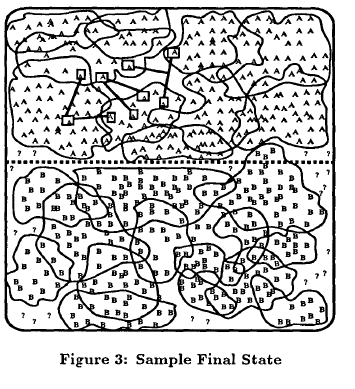
Train a supervised classifier on the labeled examples from step 2. As mentioned in earlier sections, the Yarowsky algorithm uses decision list as the base classifier, which is ranked by the purity of the distribution. Apply the resulting classifier to the entire sample set, i.e. label all examples, and add the labels where the supervised classifier was highly confident, i.e. labels above threshold, to the growing seed sets (see Figure 2).

Step4:

When the training parameters are held constant and the algorithm converges to a stable residual set, stop. Most training examples will exhibit multiple collocations indicative the same sense as shown in Figure 3, however, the decision list algorithm ensures that we only take the most reliable piece of evidence instead of a combination of all matching collocations. This strategy effectively avoided many problems and conflicts.

Setp5:

Use the previously defined optimal classifier to annotate the original untagged corpus with sense tags and probabilities or to classify to new data.



1. Princeton WordNet (Miller 1990). <https://wordnet.princeton.edu/>
2. Word Sense Disambiguation (Agirre, Eneko, Edmonds, Philip (Eds.) 2007). <https://www.springer.com/gp/book/9781402048081>
3. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods (David Yarowsky) <https://www.aclweb.org/anthology/P95-1026>