Introduction to Artificial Intelligence (236501)

Spring 2011

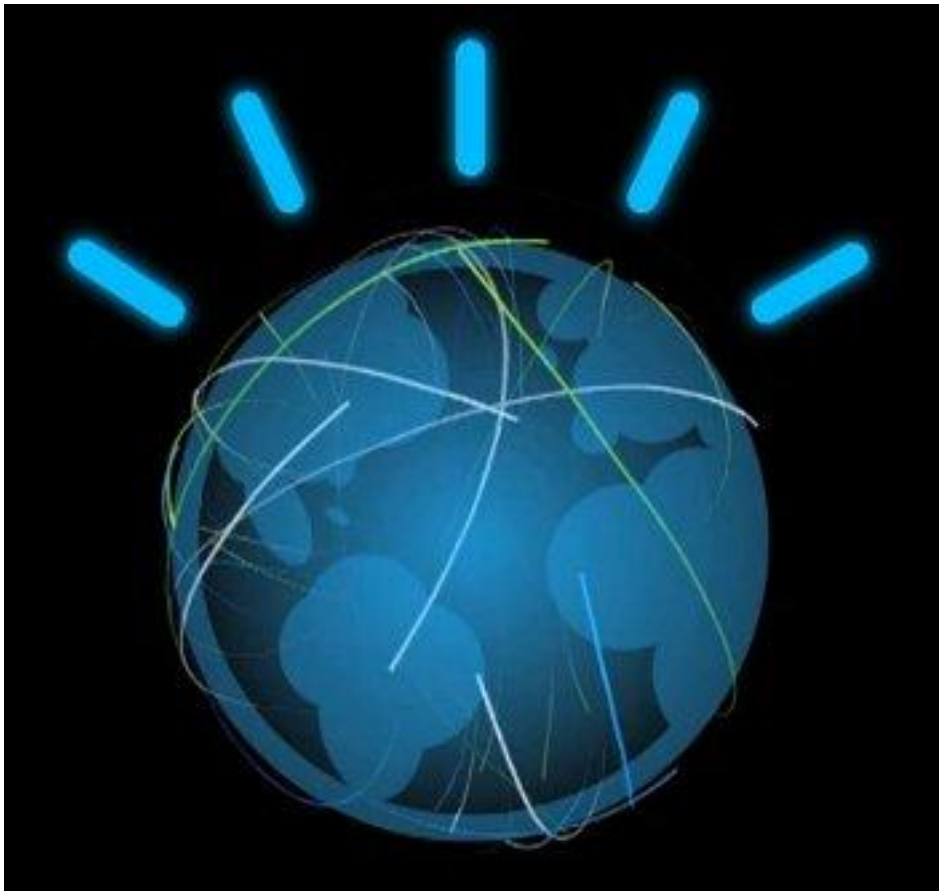
Homework Assignment 3: Text Categorization

Assignment Goals:

- To understand the impact of feature extraction on learning and classification.
- To learn basic notions in natural language processing.

Notes:

- Due date: 26/6/2011.
- This assignment is for submission in pairs or singles. No trios will be allowed.
- **Copying of any kind will not be tolerated, and will result in expulsion from the course.**
- Feel free to ask questions via email: assafgr@cs.technion.ac.il



Problem Setting:

Text categorization, also known as text classification, is the task of assigning predefined categories to free-text documents. The learning task can be framed as follows: given training documents together with their category label, return a hypothesis function (a model) that can predict the label of a document. In this setup, the instance space includes the set of all available documents. The feature space is the set of descriptors specified by the learner's background knowledge. For example, spam filters are used to detect unsolicited emails and prevent these messages from reaching a user's mailbox. The instance space is the set of all messages, and the concept is defined as the set of spam messages.

As for the feature space, text documents are usually represented as a bag-of-words (BoW)[1]. The basic idea is to represent a document in a word-based feature space, i.e., a high-dimensional vector where each dimension corresponds to a word. This process of mapping documents to feature vectors is called feature extraction. The primary goal of this assignment is to discuss the impact of feature extraction techniques on learning and classification in the context of text categorization problems. The high-dimensional feature space of text data would have also encouraged us to discuss efficient methods for feature selection, allowing the model to provide improved classifications by removing irrelevant features (noise).

During the feature extraction procedure, each word is associated with a numerical value (its corresponding feature) according to a predefined weighting scheme. Most weighting schemes are borrowed from the field of information retrieval. In this assignment we will explore the well-known TF/IDF weighting scheme. The TF/IDF is:

$$w_{fd} = tf_{fd} log\left(\frac{N}{df_f}\right)$$

Where w_{fd} is the weight of feature f in a document d, tf_{fd} is the occurrence frequency of feature f in document d, N is the total number of documents in the training data, and df_f is the number of documents contacting the feature f. The expression $log\left(\frac{N}{df_f}\right)$ is also called the *inverse document frequency* of a term (word), and denoted by idf_f. The intuition is that terms that are rare in the corpus (the instance space) better distinguish the relevant documents from the irrelevant ones. However, this simple and elegant weighting scheme still suffers from several drawbacks:

[1] More sophisticated techniques for text representation can be found in studies on Natural Language Processing (NLP).

(1) Linguistic challenges. Linguistic processing is required in the feature extraction procedure for good retrieval performance. This processing includes: case folding (e.g., The -> the), stemming (e.g., caused -> cause), morphological analysis (e.g., single -> plural), stop-word elimination (e.g., removing is, a, an, etc.). In this assignment you will analyze the effect of simple processing methods on the classification performance.
(2) Curse of dimensionality. The huge amount of words in a text corpus creates noisy data. For that reason, several applications use the idf_f as a parameter for feature selection: a threshold is set on the maximum value of the idf_f parameter, and features associated with low-frequency words are eliminated. In this assignment you will analyze the effect of different idf_f thresholds on the classification performance.

In this assignment you will deal with the problem of text categorization. You will be provided with a set of 5 text categorization problems. Each problem contains a training set of text comments associated with different topics (one topic per comment). Example topics are "*performance of Toyota Camry*" and "*sound quality of iPod nano*", etc. The goal is to return a nearest neighbor classifier (NN) that can predict the topic of a given comment. The classification results should be evaluated using hold-one-out cross-validation (HOO). For your convenient, you will be given code for the NN classifier and HOO procedure. Your challenge will be to extract and select the most relevant text features for classification.

Since only 5 learning tasks are considered, we will provide you with the code of the non-parametric McNemar's test to determine whether two classifiers produce a significantly different result on a given learning task. In this way you will be able to conclude pros and cons for pairs of classifiers on each learning task.

Assignment Instructions:

1. **Feature Extraction**

   Improve the feature extraction by implementing the following enhancements:

   a. Suffix stemming[2] – Remove/change suffixes as follows:

      1) SSES → SS
      2) IES → I
      3) SS → SS
      4) S → NULL
      5) (m>0) EED → EE (m is the number of letters before EED)
      6) ED → NULL
      7) ING → NULL
      8) AT → ATE
      9) BL → BLE
      10) IZ → IZE

      Important note: For each word, only one match is obeyed, and this will be the one with the longest matching. For example, CARESSES maps to CARESS since SSES is the longest match.

   b. Stop-word elimination – Remove the following stop words from text data: "a", "an", "and", "are", "as", "at", "be", "but", "by", "for", "if", "in", "into", "is", "it", "no", "not", "of", "on", "or", "such", "that", "the", "their", "then", "there", "these", "they", "this", "to", "was", "will", "with".

   You should analyze the effect of stemming and stop-word elimination on the estimated accuracy results. Use HOO procedure, combined with the McNemar's test, on the 5 learning tasks to support your conclusions. Note that case folding already exists in the code. Use only the provided nearest neighbor algorithm for classification.

2. **Feature Selection**

   Improve accuracy results by eliminating irrelevant features. For each learning task, select the optimal idf_f threshold for feature selection (use HOO for this purpose). You should analyze the effect of feature selection on the accuracy results. Use learning curves, with respect to changes in thresholds, to support your conclusions (i.e. accuracy as a function of the number of features).

3. **Competition**

   Devise a way to extract and select meaningful features from raw text. Explain why you have chosen this method, and why it is suitable for the task of text categorization. Your solution may be a modified version of the methods mentioned above, or, alternatively, a different method that yields better results. Compare your solution against the methods mentioned above.

---

[2] The stemming procedures you are required to implement are part of the Porter stemming algorithm. Note that you have to implement **only** the steps described in this assignment.

4. **Conclusions**
   What conclusions can you draw from these results? Do your findings reflect inherent properties of the algorithms or those of the datasets?

Important note: Since this assignment is relatively short, we shall not consider time a parameter. However, you should place an upper bound of 10 minutes on learning time and 2 seconds for classification. These bounds will be used in the competition as well.

Provided Code:

The following code is provided for your convenience:

- `learning_agent.py` – An interface for learning agents. You should inherit from this class, and override the methods:
  - `createFeatureExtractor()` – To initialize your feature extractor.
  - `createClassifier()` – Always return nearest neighbor.
  - `__str__()` – Give your agent a cool name.
- `classifier.py` – An interface for learning algorithms.
- `nearest_neighbor.py` – An implementation of the nearest neighbor algorithm.
- `feature_extractor.py` – An interface for feature extraction algorithms.
- `bag_of_words.py` – An implementation of the bag of words feature extractor.
- `agent_comparator.py` – A test-bed for comparing two agents.
- `dataset_builder.py` – A utility for creating datasets from the data files.
- `text_example.py` – An example showing how to use the provided code.
- `topics` – A folder containing the data, each category in a separate file, each instance in a separate row.

You are given five datasets, each one a combination of different topics. These datasets can be created using `createRealDatasets()` in `text_example.py`.

Competition:

In accordance with the conclusions from your research report, you will implement a learning agent for the task of text categorization. This agent will compete against agents designed by your peers on a number of undisclosed test sets. Each test set will reward the agent with a score proportional to its relative success on that set. The overall score is the sum of all scores. Winners of the competition will be awarded bonus points to their final grade.

Submission Instructions:

Dry: Research Report

- This is the most important part; it will determine your grade.
- Limit your report to **five pages**.
- The report should contain the following sections:
  - Feature Extraction Analysis
    - Results
    - Conclusions
  - Feature Selection Analysis
    - Results
    - Conclusions
  - Competition Analysis
    - Proposed Solution
    - Results
    - Conclusions
- Refrain from writing code in the report.
- Submit the printed report to the course's inbox next to the coffee cart.
- Submit an electronic version of your report and all **experimentation code** to the "Research Report" assignment in the course website.

Wet: Competition

- You will write your agent in the Python programming language. The agent must inherit from the `LearningAgent` class in the provided code.
- No use of network resources is allowed. Any attempt of sabotage or cheating will be dealt with harshly.
- You will submit (electronically) a zip file containing:
  - `submissions.txt` – Name, ID, email (comma delimited, each student in a separate row)
  - `agent.py` – Contains the implemented agent. Must contain exactly one class that inherits from LearningAgent.
  - Other Python files, in the same directory (no sub-directories).
- Make sure that your code assumes a flat hierarchy; i.e. that the code provided by the staff is in the same directory as yours.
- The submission should **not** contain any of the provided code.
- Submit the **competition code** the "Competition" assignment in the course website.

Good luck!