Recognizing web traces of various web services

**Phase 1**
Identifying loaded website

- **Features:** Each network capture is represented as a 1514-dimensional vector $x_i$.
- $x_i$: number of packets of length $i$.
- **Distribution of packet length depends on website loaded:**
  - Google
  - Wikipedia

**Conclusion:** Excellent performance on differentiating between 7 of the most popular websites.

**Dataset:** 100 training examples for each of the following websites:
- Google, Wikipedia, Bing, Youtube, Amazon, Facebook, Yahoo

**Method:**
Naive Bayes multiclass classifier with multinomial event model.

**Testing:**
5-fold Cross Validation on this dataset yields average error: 0.14%.

**Phase 2**
Locating the user action

- **Goal:** Finding the traces corresponding to auto-completion usage.
- **Training set:**
  - 200 traces of auto-completion usage
  - 20,000 packets of standard web traffic (no auto-completion)
- **Test set:**
  - 200,000 packets of "mixed" web traffic (random websites + auto-completion)
- **Algorithm:** Naive Bayes with multinomial event model.
- Sliding window to consider $n$ (≥ 20) packets at a time.
- Probability of being an auto-completion trace ($p(y)$) can vary in order to change precision and recall.

**Conclusion:** Low precision rate, especially for Bing. Could be improved by further filtering the result using the classifier derived in step 1.

Together, these two phases would allow us to scan through large webtraces and locate the data containing information about the usage of auto-completion features. Applying previous work involving side-channel attacks, we would then be able to retrieve the words searched for, using only the captured packet sizes.