Password classification

Tiko Huizinga
Supervisor: Zeno Geradts, Nederlands Forensisch Instituut (NFI)

Example case

- Police confiscates hard drives
- Fast (automatic) analysis of data needed
- Saved plain text passwords can be very useful

NFI developed forensic search engine for digital investigation

News item | 14-10-2015 | 13:35

The average amount of digital evidence found by investigators in criminal cases doubles every fifteen months. In order to continue to make these increasing amounts of data rapidly and easily searchable, the Netherlands Forensic Institute (NFI) developed the forensics search engine Hansken.

The Dutch National Police put Hansken into use in October. The police are now able to continue to carry out digital forensic investigations rapidly and efficiently. It no longer matters whether the investigators have to go through several laptops or entire server rooms. The seized data are copied in Hansken, after which the software identifies as much evidence as possible.

Because of its design, Hansken is able to recognise huge amounts of data and makes it searchable, just like a search machine. This is necessary, for the average amount of data found, for instance, in child pornography is equal to a queue of lorries five kilometres long, fully loaded with sheets printed on both sides.

Hansken

- Search engine for Dutch police and forensic institute
- Machine learning and image classification
- No password classification yet
  - This is where my research jumps in

Research question

- How can software be used to classify whether a string is a password or a *“normal”* word?

Scope

- The input for the tool are text files containing one or multiple words
- A word is the string between a starting and ending space or newline
- As a result, the tool does not classify passwords containing a space
- English language is used for training the tool

Method

- Gather data
  - Password list
  - Word list
- Generate statistics
  - Length, #Digits, #Special characters, …
- Create naive probabilistic classification tool
- Use machine learning to create classification tool
  - Support Vector Machine (SVM)
- Evaluate both tools
  - Precision, Accuracy, F1-Score

Data gathering

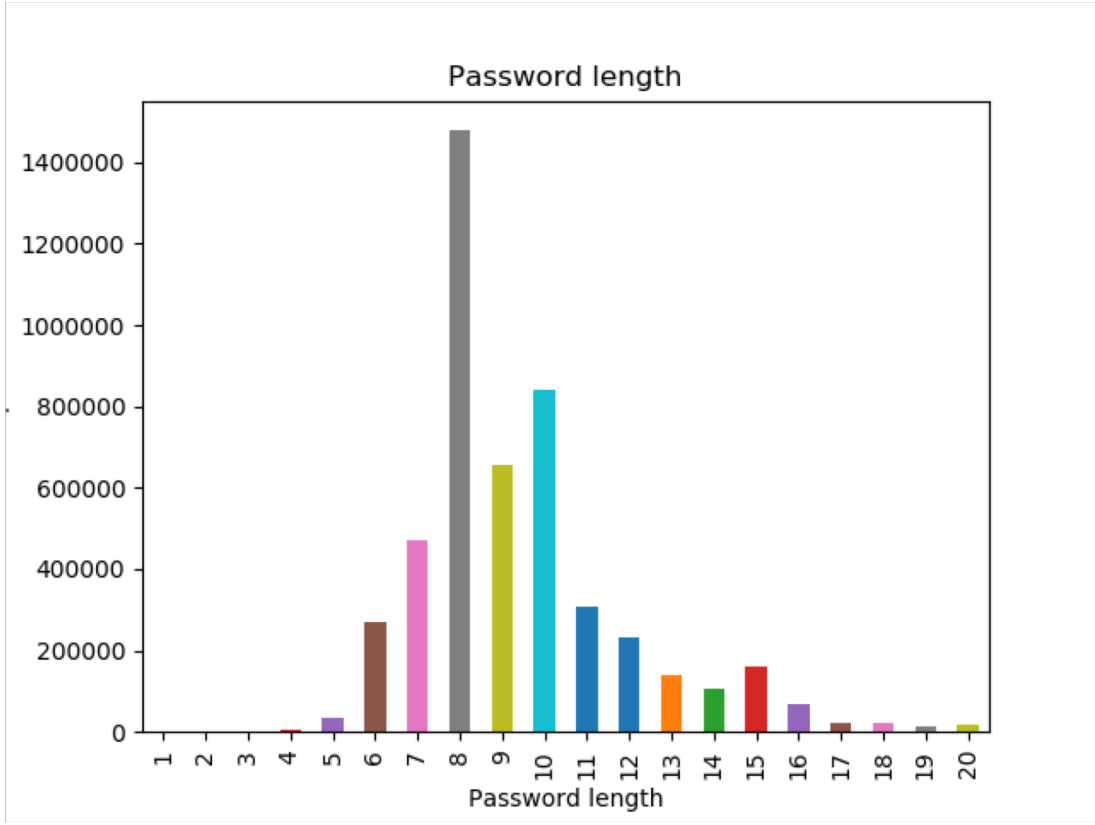
- Started with
  - Common credential list
  - English dictionary wordlist
- Too ‘boring’
  - Not a lot of special characters and no unique passwords
- New password list
  - Breach compilation
  - Unique passwords
- New word list
  - Partial Wikipedia dump
  - Represents text files on computers

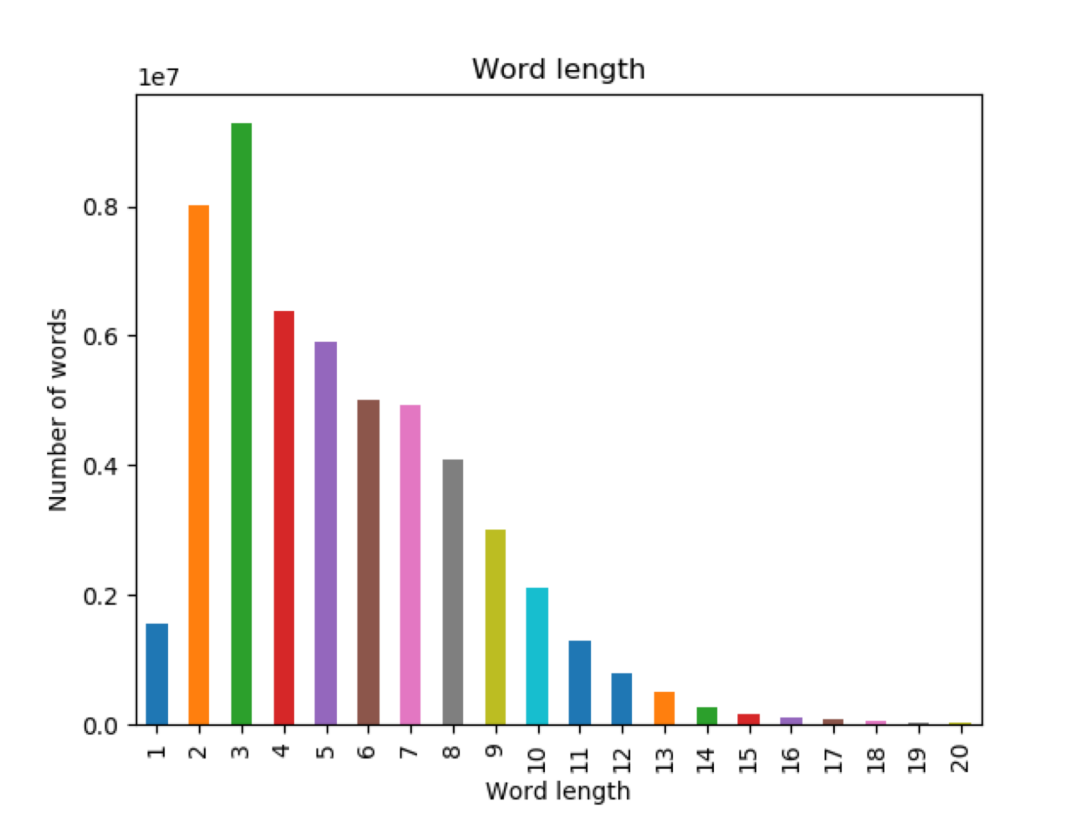
Common passwords	English wordlist
123456	abac
password	abaca
12345678	abacay
qwerty	abacas

Generate statistics

- Gather characteristics for all words
  - Length
  - # Special characters
  - # Digits
  - # Capital letters
  - # Small letters

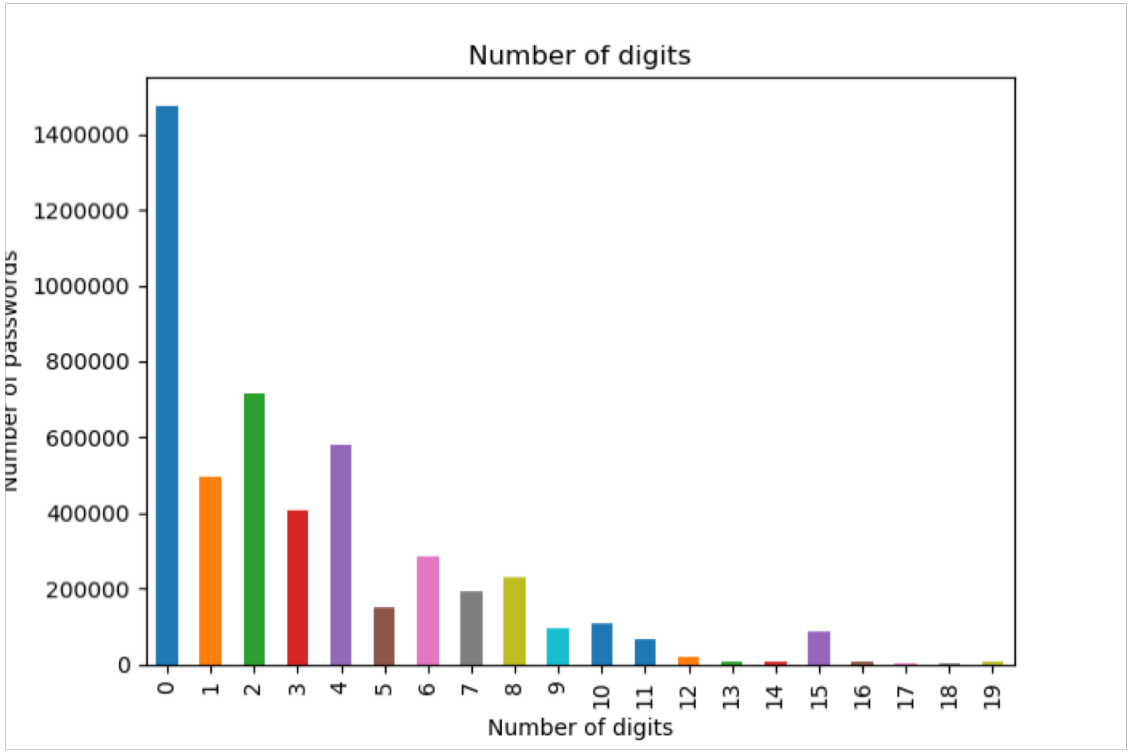
Length of passwords and words



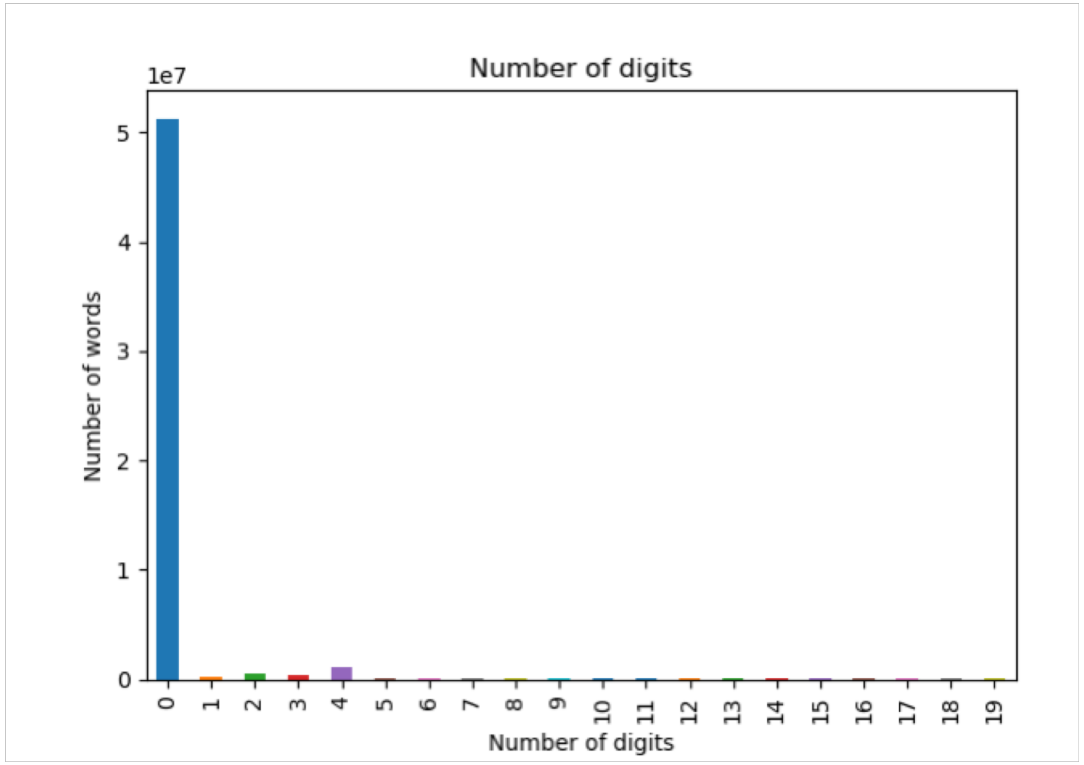


Number of digits

Passwords



Words



Naive probabilistic classifier

Class C = {Password, Word}

Characteristics X = {Length, #Special characters, #Digits, #Capital letters, #Small letters}

pw(x) = Number of passwords with characteristic x / total number of passwords

w(x) = Number of words with characteristic x / total number of words

Naive probabilistic classifier

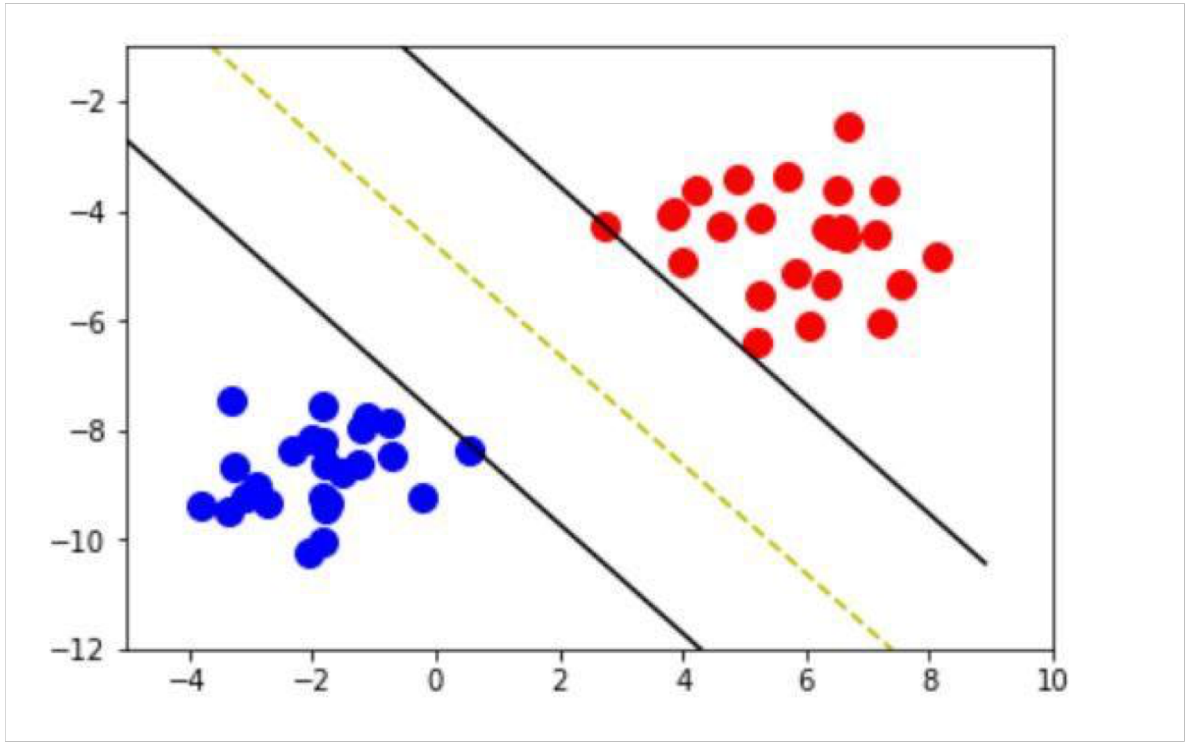
$$P(C = Password|x \epsilon X) = \frac{pw(x)}{pw(x) + w(x)}$$

$$P(C = Password|x_1, ..., x_n \epsilon X) = \frac{P(C = password|x_1) + ... + P(C = password|x_n)}{n}$$

- If result >= 0.5
  - Classify as password
- Else
  - Classify as word

Support Vector Machine (SVM)

- Machine learning classification
- Divide data in two classes
- Find hyperplane with largest margin



Metrics and evaluation of classifiers

Confusion matrix

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Metrics and evaluation of classifiers

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Metrics and evaluation of classifiers

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Metrics and evaluation of classifiers

- F1 score
- The harmonic mean of Precision and Recall

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Evaluation of classifiers

Naive probabilistic classifier

Class	Precision	Recall	F1-score
Word	0.93	0.89	0.91
Password	0.89	0.93	0.91

SVM

Class	Precision	Recall	F1-score
Word	0.79	0.91	0.85
Password	0.89	0.74	0.80

Conclusion

- *How can software be used to classify whether a string is a password or a “normal” word?*
  - A naive probabilistic classifier achieves good results with an F1 score of 0.91
  - A Support Vector Machine trains slower and achieves a lower F1 score with 0.80 and 0.85

Discussion

- The results are very dependant on the training set and test set
- SVM probably scores worse because there is no clear line separating passwords from words
- I used lists with all unique words with all the same weight
  - Giving more frequent words a higher weight might bring the model closer to reality

Future work

- Use more characteristics
  - Place of special characters in string
- Use different (machine learning) classification algorithms
  - Decision trees
  - Bayesian networks
  - SVM with different parameters

Thank you!