Cipher Type Detection

Malte Nuhn
Human Language Technology
and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
nuhn@cs.rwth-aachen.de

Kevin Knight
Information Sciences Institute
University of Southern California
knight@isi.edu

Abstract

Manual analysis and decryption of enciphered documents is a tedious and error prone work. Often—even after spending large amounts of time on a particular cipher—no decipherment can be found. Automating the decryption of various types of ciphers makes it possible to sift through the large number of encrypted messages found in libraries and archives, and to focus human effort only on a small but potentially interesting subset of them. In this work, we train a classifier that is able to predict which encipherment method has been used to generate a given ciphertext. We are able to distinguish 50 different cipher types (specified by the American Cryptogram Association) with an accuracy of 58.5%. This is a 11.2% absolute improvement over the best previously published classifier.

1 Introduction

Libraries and archives contain a large number of encrypted messages created throughout the centuries using various encryption methods. For the great majority of the ciphers an analysis has not yet been conducted, simply because it takes too much time to analyze each cipher individually, or because it is too hard to decipher them. Automatic methods for analyzing and classifying given ciphers makes it possible to sift interesting messages and by that focus the limited amount of human resources to a promising subset of ciphers.

For specific types of ciphers, there exist automated tools to decipher encrypted messages. However, the publicly available tools often depend on a more or less educated guess which type of encipherment has been used. Furthermore, they often still need human interaction and are only restricted to analyzing very few types of ciphers. In practice however, there are many different types of ciphers which we would like to analyze in a fully automatic fashion: Bauer (2010) gives a good overview over historical methods that have been used to encipher messages in the past. Similarly, the American Cryptogram Association (ACA) specifies a set of 56 different methods for enciphering a given plaintext:

Each encipherment method M_i can be seen as a function that transforms a given plaintext into a ciphertext using a given key, or short:

$$\text{cipher} = M_i(\text{plain}, \text{key})$$

When analyzing an unknown ciphertext, we are interested in the original plaintext that was used to generate the ciphertext, i.e. the opposite direction:

$$\text{plain} = M_i^{-1}(\text{cipher}, \text{key})$$

Obtaining the plaintext from an enciphered message is a difficult problem. We assume that the decipherment of a message can be separated into solving three different subproblems:

1. Find the encipherment method M_i that was used to create the cipher
$$\text{cipher} \quad \rightarrow \quad M_i$$
2. Find the key that was used together with the method M_i to encipher the plaintext to obtain $cipher = M_i(plain, key)$.
3. Decode the message using M_i and key
$$\text{cipher} \quad \rightarrow \quad M_i^{-1}(\text{cipher}, \text{key})$$

Thus, an intermediate step to deciphering an unknown ciphertext is to find out which encryption method was used. In this paper, we present a classifier that is able to predict just that: Given an unknown ciphertext, it can predict what kind of encryption method was most likely used to generate

- Type: CMBIFID
- Plaintext:
  WOMEN NSFOO TBALL ISGAI
  NINGI NPOPU LARIT YANDT
  HETOU RNAME
- Key:
  LEFTKEY='IACERATIONS'
  RIGHTKEY='KNORKOPPING'
  PERIOD=3, LROUTE=1
  RROUTE=1, USE6X6=0
- Ciphertext:
  WTQNG GEEBQ BPNQP VANEN
  KDAOD GAHQS PKNVI PTAAP
  DGMGR PCSGN

Figure 1: Example "CMBIFID" cipher: Text is grouped in five character chunks for readability.

it. The results of our classifier are a valuable input to human decipherers to make a first categorization of an unknown ciphertext.

2 Related Work

Central to this work is the list of encryption methods provided by the American Cipher Association[1]. This list contains detailed descriptions and examples of each of the cipher types, allowing us to implement them. Figure 3 lists these methods.

We compare our work to the only previously published cipher type classifier for classical ciphers[2]. This classifier is trained on $16,800$ ciphertexts and is implemented in javascript to run in the web browser: The user can provide the ciphertext as input to a web page that returns the classifier's predictions. The source code of the classifier is available online. Our work includes a reimplementation of the features used in that classifier.

As examples for work that deals with the automated decipherment of cipher texts, we point to (Ravi and Knight, 2011), and (Nuhn et al., 2013). These publications develop specialized algorithms for solving simple and homophonic substitution ciphers, which are just two out of the 56 cipher types defined by the ACA. We also want to mention (de Souza et al., 2013), which presents a cipher type classifier for the finalist algorithms of the Advanced Encryption Standard (AES) contest.

[1] http://cryptogram.org/cipher_types.html

[2] See http://bionsgadgets.appspot.com/gadget_forms/refscore_extended.html and https://sites.google.com/site/bionspot/cipher-id-tests

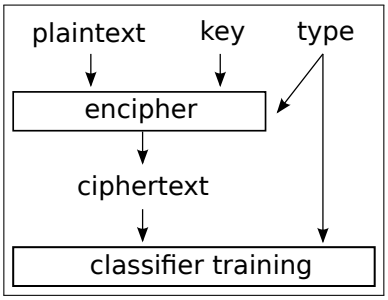

Figure 2: Overview over the data generation and training of the classifier presented in this work.

3 General Approach

Given a ciphertext, the task is to find the right encryption method. Our test set covers 50 out of 56 cipher types specified by ACA, as listed in Figure 3. We are going to take a machine learning approach which is based on the observation that we can generate an infinite amount of training data.

3.1 Data Flow

The training procedure is depicted in Figure 2: Based upon a large English corpus, we first choose possible plaintext messages. Then, for each encipherment method, we choose a random key and encipher each of the plaintext messages using the encipherment method and key. By doing this, we can obtain (a theoretically infinite) amount of labeled data of the form (type, ciphertext). We can then train a classifier on this data and evaluate it on some held out data.

Figure 1 shows that in general the key can consist of more than just a codeword: In this case, the method uses two codewords, a period length, two different permutation parameters, and a general decision whether to use a special "6×6" variant of the cipher or not. If not defined otherwise, we choose random settings for these parameters. If the parameters are integers, we choose random values from a uniform distribution (in a sensible range). In case of codewords, we choose the 450k most frequent words from an English dictionary. We train on cipher texts of random length.

3.2 Classifiers

The previous state-of-the-art classifier by BION uses a random forest classifier (Breiman, 2001). The version that is available online, uses 50 ran-

- 6x6bifid
- 6x6playfair
- amsco
- bazeries
- beaufort
- bifid6
- bifid7
- (cadenus)
- cmbifid
- columnar
- digrafid
- dbl chckrbrd
- four square
- fracmorse
- grandpre
- (grille)
- gromark
- gronsfeld
- homophonic
- mnmedinome
- morbit
- myszkowski
- nicodemus
- nihilistsub
- (nihilisttransp)
- patristocrat
- period 7 vig.
- periodic gromark
- phillips
- plaintext
- playfair
- pollux
- porta
- portax
- progkey beaufort
- progressivekey
- quagmire2
- quagmire3
- quagmire4
- ragbaby
- randomdigit
- randomtext
- redefence
- (route transp)
- runningkey
- seriatedpfair
- swagman
- tridigital
- trifid
- trisquare
- trisquare hr
- two square
- two sq. spiral
- vigautokey
- (vigenere)
- (vigslidefair)

Figure 3: Cipher types specified by ACA. Our classifier is able to recognize 50 out of these 56 ciphers. The braced cipher types are not covered in this work.

dom decision trees. The features used by this classifier are described in Section 4.

Further, we train a support vector machine using the libSVM toolkit (Chang and Lin, 2011). This is feasible for up to 100k training examples. Beyond this point, training times become too large. We perform multi class classification using ν-SVC and a polynomial kernel. Multi class classification is performed using one-against-one binary classification. We select the SVM's free parameters using a small development set of 1k training examples.

We also use Vowpal Wabbit (Langford et al., 2007) to train a linear classifier using stochastic gradient descent. Compared to training SVMs, Vowpal Wabbit is extremely fast and allows using a lot of training examples. We use a squared loss function, adaptive learning rates and don't employ any regularization. We train our classifier with up to 1M training examples. The best performing settings use one-against-all classification, 20 passes over the training data and the default learning rate. Quadratic features resulted in much slower training, while not providing any gains in accuracy.

4 Features

We reimplemented all of the features used in the BION classifier, and add three newly developed sets of features, resulting in a total of 58 features.

In order to further structure these features, we group these features as follows: We call the set of features that relate to the length of the cipher `LEN`. This set contains binary features firing when the cipher length is a multiple of 2, 3, 5, 25, any of 4-15, and any of 4-30. We call the set of features that are based on the fact that the ciphertext contains a specific symbol `HAS`. This set contains binary features firing when the cipher contains a digit, a letter (A-Z), the "#" symbol, the letter "j", the digit "0". We also introduce another set of features called `DGT` that contains two features, firing when the cipher is starting or ending with a digit. The set `VIG` contains 5 features: The feature score is based on the best possible bigram LM perplexity of a decipherment compatible with the decipherment process of the cipher types Autokey, Beaufort, Porta, Slidefair and Vigenere. Further, we also include the features `IC`, `MIC`, `MKA`, `DIC`, `EDI`, `LR`, `ROD` and `LDI`, `DBL`, `NOMOR`, `RDI`, `PTX`, `NIC`, `PHIC`, `BDI`, `CDD`, `SSTD`, `MPIC`, `SERP`, which were introduced in the BION classifier[3]. Thus, the first 22 data points in Figure 4 are based on previously known features by BION. We further present the following additional features.

4.1 Repetition Feature (`REP`)

This set of features is based on how often the ciphertext contains symbols that are repeated exactly n times in a row: For example the ciphertext shown in Figure 1 contains two positions with repetitions of length $n = 2$, because the ciphertext contains EE, as well as AA. Beyond length 2, there are no repeats. These numbers are then normalized by dividing them by the total number of repeats of length $2 \leq n \leq 5$.

4.2 Amsco Feature (`AMSC`)

The idea of the AMSCO cipher is to fill consecutive chunks of one and two plaintext characters into n columns of a grid (see Table 1). Then a permutation of the columns is performed, and the resulting permuted plaintext is read of line by line and forms the final ciphertext. This feature reads the ciphertext into a similar grid of up to 5 columns

[3] See `http://home.comcast.net/~acabion/acarefstats.html`

Plaintext	w oo	om t	e ba	ns l	f li
Permutation	3	5	1	4	2

Table 1: Example grid used for AMSCO ciphers.

and then tries all possible permutations to retain the original plaintext. The result of this operation is then scored with a bigram language model. Depending on whether the difference in perplexity between ciphertext and deciphered text exceeds a given threshold, this binary feature fires.

4.3 Variant Feature (VAR)

In the variant cipher, the plaintext is written into a block under a key word. All letters in the first column are enciphered by shifting them using the first key letter of the key word, the second column uses the second key letter, etc. For different periods (i.e. lengths of key words), the ciphertext is structured into n columns and unigram statistics for each column are calculated. The frequency profile of each column is compared to the unigram frequency profile using a perplexity measure. This binary feature fires when the resulting perplexities are lower than a specific threshold.

5 Results

Figure 4 shows the classification accuracy for the BION baseline, as well as our SVM and VW based classifiers for a test set of 305 ciphers that have been published in the ACA. The classifiers shown in this figure are trained on cipher texts of random length. We show the contribution of all the features we used in the classifier on the x-axis. Furthermore we also vary the amount of training data we use to train the classifiers from 10k to 1M training examples. It can be seen that when using the same features as BION, our prediction accuracy is compatible with the BION classifier. The main improvement of our classifier stems from the REP, AMSC and VAR features. Our best classifier is more than 11% more accurate than previous state-of-the-art BION classifier.

We identified the best classifier on a held-out set of 1000 ciphers, i.e. 20 ciphers for each cipher type. Here the three new features improve the VW-1M classifier from 50.9% accuracy to 56.0% accuracy, and the VW-100k classifier from 48.9% to 54.6%. Note that this held-out set is based on the exact same generator that we used to create the training data with. However, we also report the results of our method on the completely independently created ACA test set in Figure 4.

6 Conclusion

We presented a state-of-the art classifier for cipher type detection. The approach we present is easily extensible to cover more cipher types and allows incorporating new features.

Acknowledgements

We thank Taylor Berg-Kirkpatrick, Shu Cai, Bill Mason, Beáta Megyesi, Julian Schamper, and Megha Srivastava for their support and ideas. This work was supported by ARL/ARO (W911NF-10-1-0533) and DARPA (HR0011-12-C-0014).

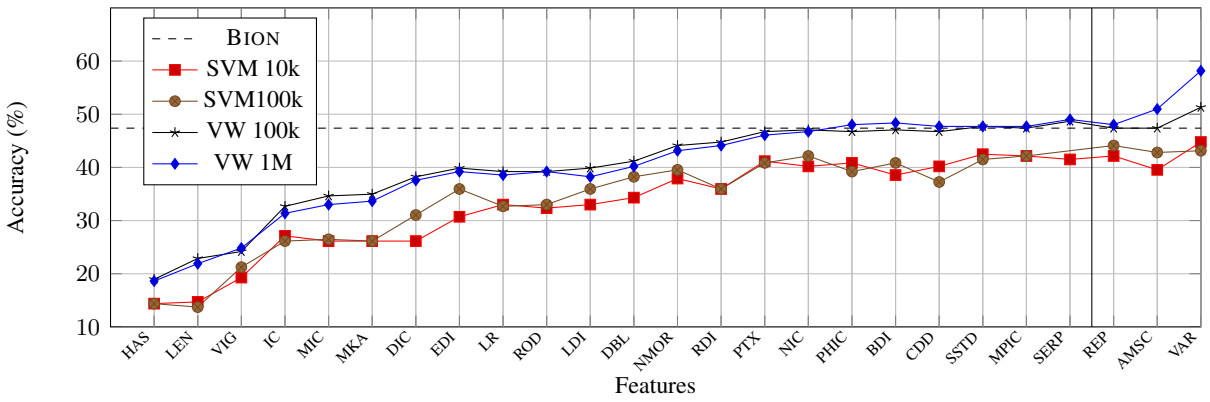

Figure 4: Classifier accuracy vs. training data and set of features used. From left to right more and more features are used, the x-axis shows which features are added. The feature names are described in Section 4. The features right of the vertical line are presented in this paper. The horizontal line shows the previous state-of-the art accuracy (BION) of 47.3%, we achieve 58.49%.

References

F.L. Bauer. 2010. *Decrypted Secrets: Methods and Maxims of Cryptology*. Springer.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32, October.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

William AR de Souza, Allan Tomlinson, and Luiz MS de Figueiredo. 2013. Cipher identification with a neural network.

John Langford, Lihong Li, and Alex Strehl. 2007. Vowpal Wabbit. `https://github.com/JohnLangford/vowpal_wabbit/wiki`.

Malte Nuhn, Julian Schamper, and Hermann Ney. 2013. Beam search for solving substitution ciphers. In *ACL (1)*, pages 1568–1576.

Sujith Ravi and Kevin Knight. 2011. Bayesian Inference for Zodiac and Other Homophonic Ciphers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 239–247, Stroudsburg, PA, USA, June. Association for Computational Linguistics.