Matching Keys and Encrypted Manuscripts

Eva Pettersson and Beáta Megyesi
Department of Linguistics and Philology
Uppsala University
firstname.lastname@lingfil.uu.se

Abstract

Historical cryptology is the study of historical encrypted messages aiming at their decryption by analyzing the mathematical, linguistic and other coding patterns and their historical context. In libraries and archives we can find quite a lot of ciphers, as well as keys describing the method used to transform the plaintext message into a ciphertext. In this paper, we present work on automatically mapping keys to ciphers to reconstruct the original plaintext message, and use language models generated from historical texts to guess the underlying plaintext language.

1 Introduction

Hand-written historical records constitute an important source, without which an understanding of our society and culture would be severely limited. A special type of hand-written historical records are encrypted manuscripts, so called ciphers, created with the intention to keep the content of the message hidden from others than the intended receiver(s). Examples of such materials are political, diplomatic or military correspondence and intelligence reports, scientific writings, private letters and diaries, as well as manuscripts related to secret societies.

According to some historians' estimates, one percent of the material in archives and libraries are encrypted sources, either encrypted manuscripts called ciphertexts, decrypted or original plaintext, and/or keys describing how the encryption/decryption is performed. The manuscripts are usually not indexed as encrypted sources (Láng, 2018), which makes it difficult to find them unless you are lucky to know a librarian with extensive knowledge about the selection of the particular library you are digging in. In addition, related ciphertexts, plaintexts and keys are usually not stored together, as information about how the encrypted sources are related to each other is lost. If the key has not been destroyed over time – unintentionally, or intentionally for security reasons – it is probably kept in a different place than the corresponding ciphertext or plaintext given that these were probably produced in different places by different persons, and eventually ended up in different archives. Information about the origin of the ciphertext and key, such as dating, place, sender and/or receiver, or any cleartext in the manuscript, might give some important clues to the probability that a key and a ciphertext originate from the same time, and persons. However, information about metadata is far from enough to link the related encrypted sources to each other. It is a cumbersome, if not impossible, process for a historian to try to map a bunch of keys to a pile of ciphertexts scattered in the archive in order to try to decrypt these on the basis of the corresponding key. Further, the cryptanalyst might reconstruct a key given a ciphertext, and the reconstructed key might be applicable to other ciphertexts as well thereby providing more decrypted source material.

In this paper, we present work on automatically mapping ciphertext sequences to keys to return the plaintext from the ciphertext based on simple and homophonic substitution from Early Modern times. We measure the output of the mapping by historical language models developed for 14 European languages to make educated guesses about the correct decryption of ciphertexts. The method is implemented in a publicly available online user interface where users can upload a transcribed key and a ciphertext and the tool returns the plaintext output along with a probability measure of how well the decrypted plaintext matches historical language models for these European languages.

In Section 2, we give a brief introduction to historical cryptology with the main focus on en-

crypted sources and keys. Section 3 describes the method for mapping ciphers to their corresponding keys. The experimental results are presented in Section 4 and discussed in Section 5. Finally, we conclude the paper in Section 6.

2 Historical Cryptology

Ciphers use a secret method of writing, based on an encryption algorithm to generate a *ciphertext*, which in turn can be used to decrypt the message to retrieve the intended, underlying information, called the *plaintext*. A cipher is usually operated on the basis of a *key*. The key contains information about what output the cipher shall produce given the plaintext characters in some specific language.

Historical cryptology is the study of encoded or encrypted messages from our history aiming at the decryption by analyzing the mathematical, linguistic and other coding patterns and their histories. One of the main and glorious goals is to develop algorithms for decryption of various types of historical ciphers, i.e. to reconstruct the key in order to retrieve the corresponding plaintext from a ciphertext. The main focus for cryptanalysts has been on specific ciphers, see e.g. (Bauer, 2017; Singh, 2000) for nice summaries, while systematic decryption of various cipher types on a larger scale has been paid less attention to (see e.g. Knight et al. (2006); Nuhn and Knight (2014); Ravi and Knight (2008)). Historians, on the other hand, are searching for ciphertexts and keys in libraries to reveal new, important and hitherto hidden information to find new facts and interpretations about our history. Another, less observed goal within historical cryptology is therefore to map the encrypted sources, the original keys and corresponding ciphertexts.

There are many different types of ciphers used throughout our history (Kahn, 1996). In early modern times, when encryption became frequently used in Europe, ciphers were typically based on transposition, where the plaintext characters are reordered in a systematic way, or substitution of plaintext characters to transform each character in the plaintext to another symbol from existing alphabets, digits, special symbols, or a mixture of these (Bauer, 2007). More advanced substitution ciphers include homophonic, polygraphic, and polyalphabetic substitution. In Figure 1, we show a homophonic substitution cipher with a key, a short ciphertext and the corresponding plaintext generated by the key. Each plaintext character, written in capital letter, has one or several corresponding symbol(s) by which the plaintext characters are substituted to encrypt the message. To make decryption difficult, the most frequently occurring plaintext characters are usually substituted with one of several possible symbols.

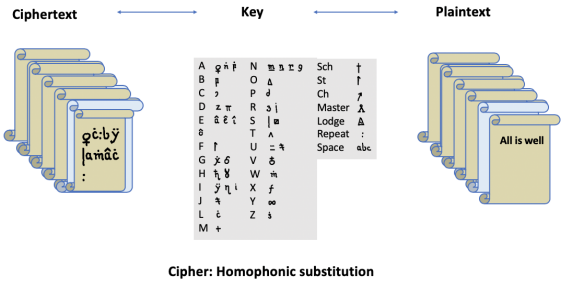

Figure 1: Ciphertext, key, and the corresponding plaintext for a homophonic substitution cipher.

Ciphertexts contain symbol sequences with spaces, or without any space to hide word boundaries. Similar to historical text, punctuation marks are not frequent, sentence boundaries are typically not marked, and capitalized initial characters in the beginning of the sentence are usually missing. We can also find nulls in ciphertexts, i.e. symbols without any corresponding plaintext characters to confuse the cryptanalyst to make decryption even harder.

Keys might contain substitution of not only characters in the plaintext alphabet, but also nomenclatures where bigrams, trigrams, syllables, morphemes, common words, and/or named entities, typically referring to persons, geographic areas, or dates, are substituted with certain symbol(s). Diacritics and double letters are usually not encoded. Each type of entity to be encrypted might be encoded by one symbol only (unigragh), two symbols (digraph), three symbols (trigraph), and so on. For example, the plaintext alphabet characters might be encrypted with codes using two-digit numbers, the nomenclatures with three-digit numbers, space with one-digit numbers, and the nulls with two-digit numbers, etc. Figure 2 illustrates a key based on homophonic substitution with nomenclature from the second half of the 17th century. Each letter in the alphabet has at least one corresponding ciphertext symbol, represented as a two-digit number (digraph), and the vowels and double consonants have one additional

graphical sign (unigraph). The key also contains encoded syllables with digraphs consisting of numerals or Latin characters, followed by a nomenclature in the form of a list of Spanish words encoded with three-digit numbers letters or graphical signs.

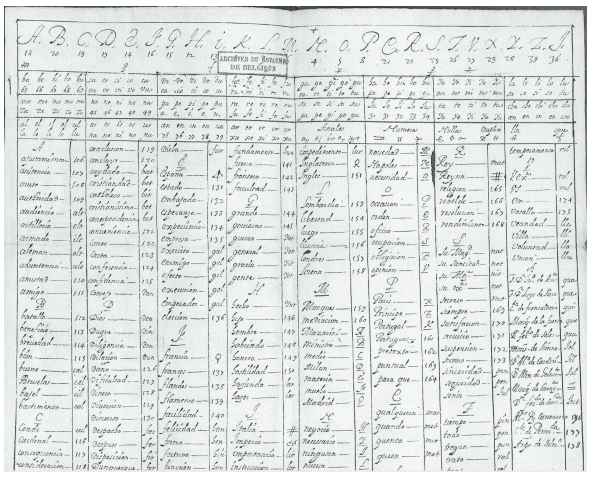

Figure 2: A key from the second half of the 17th century (Algemeen Rijksarchief, 1647-1698) from the DECODE database (Megyesi et al., 2019).

One of the first steps, apart from digitization of the encrypted source, is the transcription of keys and ciphertext images, before cryptanalysis can be applied, aiming at the decryption of the ciphertext to recover the key. However, there are other challenges that also need attention depending on what document types that are available and what documents that need to be recovered. These are:

1. generate ciphertext given a plaintext and a key (i.e. encryption)
2. reconstruct plaintext from a ciphertext and a key (less trivial due to unknown plaintext language and ambiguous code sequences and nulls)
3. map key and ciphertexts

Next, we will describe our experiments on retrieving plaintext from keys given a ciphertext with the goal to be able to automatically find ciphertexts that belong to a particular key.

3 Mapping Ciphers to Keys

3.1 Data

For our experiments on automatically mapping ciphertext sequences to key-value pairs, we need access to four kinds of input files:

1. a transcribed ciphertext
2. its corresponding key
3. its corresponding plaintext (for evaluation purposes)
4. a set of language models (for language detection)

A collection of several hundreds of ciphers and keys from Early Modern times can be found in the recently developed DECODE database (Megyesi et al., 2019).[1] The database allows anyone to search in the collection, whereas upload of new encrypted manuscripts may be done by registered users only. The collected ciphertexts and keys are annotated with metadata including information about the provenance and location of the manuscript, transcription, possible decryption(s) and translation(s) of the ciphertext, images, and any additional material of relevance to the particular manuscript.

Currently, most of the ciphertexts in the DECODE database are still unsolved. Even though the overall aim of the cipher-key mapping algorithm (CKM) is to automatically try to match these unsolved ciphertexts to existing keys, we need previously solved ciphertexts, connected to a key and a plaintext file, to conduct our experiments. In our experiments, we thus make use of three previously broken keys and their corresponding ciphertexts, written originally during Early Modern times. For all three manuscripts, the transcribed ciphertext as well as the key and a plaintext version of the contents are available through the DECODE database. We also add a fourth decrypted file, the *Copiale* cipher (Knight et al., 2011),[2] for which both the ciphertext and the plaintext are accessible through the DECODE database. This cipher will only be used for evaluation of the language detection part of the algorithm. The data collection used for the experiments is summarized in Table 1, where each manuscript is described with regard to its cipher type and plaintext language.

In our experiments, we will use the *Barb* cipher for initial tests during the development of the CKM algorithm, hence-forth the *training set*. This cipher is based on homophonic substitution with

[1] https://cl.lingfil.uu.se/decode/

[2] https://cl.lingfil.uu.se/~bea/copiale/

Name	Cipher type	Plaintext	Use
Barb.lat.6956	homophonic, nulls, nomenclature	Italian	training
Francia-64	homophonic, nulls, nomenclature	Italian	evaluation
Borg.lat.898	simple substitution	Latin	evaluation
Copiale	homophonic	German	evaluation (lang. detection)

Table 1: Datasets used for training and evaluation of the CKM algorithm.

nomenclatures, and consists of codes with numbers. The codes are 2-digit numbers representing plaintext characters, syllables and some function words, and three-digit codes for place and person names and common words. The cipher also contains two one-digit codes denoting word boundaries. The *evaluation set* on the other hand, consists of two ciphers:

1. the *Francia* cipher, which has the same cipher type (homophonic substitution with nulls and nomenclature) and underlying plaintext language (Italian) as the Barb cipher used during training

2. the more divergent *Borg* cipher, which is instead a simple substitution cipher with Latin as the underlying plaintext language

The transcription of the ciphertexts was also retrieved from the DECODE database. Each transcription file of a particular cipher (which may consist of one or multiple images) starts with comment lines (marked by "#") with information about the name of the file, the image name, the transcriber's id, the date of the transcription, etc. The transcription is carried out symbol by symbol and row by row keeping line breaks, spaces, punctuation marks, dots, underlined symbols, and cleartext words, phrases, sentences, paragraphs, as shown in the original image. In case cleartext is embedded in the ciphertext, the cleartext sequence is clearly marked as such with a language id. For a detailed description of the transcription, we refer to Megyesi et al. (2019).

Original and reconstructed keys also follow a common format, starting with metadata about the key followed by a description of the code groups. Each code is described in a separate line followed by the plaintext entity (character, syllable, word, etc), delimited by at least one space character.

Figure 3 shows a few lines from the key file belonging to the Barb cipher, with codes 1 and 8 denoting a word boundary, codes 00 and 02 denoting the letter *a*, code 03 denoting the letter *o*, code 04 denoting either the letter *u* or the letter *v* (since there was often no distinction between these two letters in historical texts), and code 232 denoting the whole phrase *ambasciatore di Spagna* (ambassador of Spain).

```
#Key
#homophonic with nomenclature
#null = space (word boundary)
1 <null>
8 <null>
00 a
02 a
03 o
04 u/v
...
232 ambasciatore di Spagna
```

Figure 3: Example format for a transcribed key file in the Decode database.

3.2 Storing code-value pairs

In the first step of the CKM algorithm, the key file is processed and the code-value pairs, as well as the length of the longest code, are stored for future reference. The key file is required to be in plain text format, and with one code-value pair on each line. Furthermore, the code and its value should be separated by at least one white space character. If the key file contains values denoting word delimiters, the script will recognise these as such if they are written as "null" (in any combination of upper-case and lower-case characters, so for example "Null" and "NULL" will also be recognised as word delimiters). Any lines initialized with a hashtag sign ("#") will be ignored, as containing comments, in accordance with the format illustrated in Figure 3.

3.3 Mapping ciphertexts to code-value pairs

In the second step, the transcribed ciphertext is processed, and its contents matched against the code-value pairs stored in the previous step, to re-

```
#002r.jpg
<IT De Inenunchi? 14 Maggio 1628.>
623967501737823323650234305182200462 3?
```

Figure 4: Example of a ciphertext segment in the Decode database.

veal the underlying plaintext message. In this process, three types of input are ignored:

1. text within angle brackets (presumed to contain cleartext segments)
2. lines starting with a hashtag (presumed to be comments)
3. question marks (presumed to denote the annotator's uncertainty about the transcription)

The first lines from a transcribed ciphertext file in the Decode database is shown in Figure 4, where the name of the image from which the transcription has been made is given as a comment (preceded by a hashtag), cleartext is given within angle brackets, and the annotator's uncertainty about transcribing the last character on the line as the digit "3" is signalled by a question mark.

If the transcribed ciphertext contains word boundaries in the form of space characters, or if the code-value pairs stored in the previous step contain codes for denoting word boundaries, the ciphertext is split into words based on this information, and the remaining mapping process is performed word by word. If no such information exists, the whole ciphertext is treated as a single segment, to be processed character by character.

If word boundaries have been detected, for every word that is shorter than, or equal in length to, the longest code in the key file, we check if the whole word can be matched towards a code. If so, we replace the ciphertext sequence with the value connected to that code. If not, or if the word is longer than the longest code in the key file, we iterate over the sequence, character by character, and try to match each character against the key file. If not successful, we merge the current character(s) with the succeeding character, and try to match the longer sequence against the key file, until we reach a sequence equal in length to the longest code in the key file. If nothing can be matched in the key file for the maximum length sequence, we replace this sequence by a question mark instead, and move on to the next character.

```
If sequence equals a code:
  Replace sequence by matched value
Else:
  While characters in sequence:
    If char(s) equals a code:
      Replace char(s) by matched value
    Else if char length equals
    longest code length:
      Replace char(s) by ? and
      move on to next character
    Else:
      Merge char with next char
      and try again
```

Figure 5: Algorithm for mapping ciphertext sequences to code-value pairs.

This non-greedy search-and-replace mechanism is applied for the whole word, except for the end of the word. Since we know that the key in the training file contains code-value pairs for representing suffixes, the script checks, in each iteration, if the remaining part of the word is equal in length to the longest code in the key file. If so, we try to match the whole sequence, and only if this fails, we fall back to the character-by-character mapping. The whole algorithm for the matching procedure is illustrated in Figure 5.

3.4 Language identification

When the plaintext has been recovered, the next task is to guess what language the decrypted text is written in, and present hypotheses to the user. This is done based on language models for historical text, derived from the HistCorp collection of historical corpora and tools (Pettersson and Megyesi, 2018).[3] We use word-based language models created using the IRSTLM package (Federico et al., 2008), for the 14 languages currently available from the HistCorp webpage: Czech, Dutch, English, French, German, Greek, Hungarian, Icelandic, Italian, Latin, Portuguese, Slovene, Spanish and Swedish. For this particular task, we only make use of the unigram models, i.e. the single words in the historical texts. The plaintext words generated in the previous steps, by matching character sequences in the ciphertext against code-value pairs in the key file, are compared to the words present in the language model for each of these 14 languages. As output, the script produces a ranked list of these languages, presenting the percentage of words in the plaintext file that are also found in the word-based language model

[3] https://cl.lingfil.uu.se/histcorp/

for the language in question. The idea is that if a large percentage of the words in the decrypted file are also present in a language model, there is a high chance that the key tested against the ciphertext is actually an appropriate key for this particular text.

3.5 Evaluation

We evaluate our method for cipher-key mapping based on the percentage of words in the evaluation corpus that are identical in the automatically deciphered text and in the manually deciphered gold standard version (taking the position of the word into consideration). Casing is not considered in the comparison, since lower-case and upper-case words are usually not represented by different codes in the key files used in our experiments. Furthermore, some codes may refer to several (related) values, such as in the example given in Figure 3 (see further Section 3.2), where the code 04 could correspond either to the letter *u* or to the letter *v*. This also holds for different inflectional forms of the same lemma, such as the Italian word for 'this', that could be *questo, questa* or *questi*, depending on the gender and number of the head word that it is connected to, and therefore would typically be represented by the same code. In these cases, we consider the automatic decipherment to be correct, if any of the alternative mappings corresponds to the form chosen in the gold standard.

The language identification task is evaluated by investigating at what place in the ranked list of 14 potential languages the target language is presented.

4 Results

4.1 Cipher-Key mapping

In the first experiment, we tested the CKM algorithm on the Francia cipher (see further Section 3.1); a cipher of the same type as the cipher used as a role model during the development of the method. This cipher shares several characteristics with the training text: (i) it is written during the same time period, (ii) it is collected from the Vatican Secret Archives, (iii) it is a numerical cipher with homophonic substitution and nomenclatures, and (iv) word delimiters are represented in the key.

When comparing the automatically decrypted words to the words in the manually deciphered gold standard, approximately 79% of the words are identical. The cases where the words differ could be categorized into four different types:

1. **Incomplete key: Diacritics**
   (36 instances)
   The key only contains plain letters, without diacritics. The human transcriber has however added diacritics in the manually deciphered text, where applicable. For example, the code 9318340841099344 has been interpreted by the script as the word *temerita*. The human transcriber has added a diacritic to the last letter *a*, resulting in the word *temerità* ('boldness'), even though the key states *a* (without accent) as the value for the code "44".

2. **Character not repeated**
   (27 instances)
   In cases where a character should be repeated in order for a word to be spelled correctly (at least according to present-day spelling conventions), the human transcriber has in many cases chosen to repeat the character, even though this is not stated in the key.

3. **Human reinterpretation**
   (2 instances)
   In a few cases where the text contains unexpected inflectional forms, such as a singular ending where a plural ending would be expected, the human transcriber has chosen the grammatically correct form, even though the code in the ciphertext actually is different.

4. **Wrong interpretation by the script**
   (16 instances)
   Due to the non-greedy nature of the algorithm, it will sometimes fail to match longer codes in the key file, when it finds a match for a shorter code. This could be seen for example for prefixes such as buon- in *buonissima* ('very good'), and qual- in *qualche* ('some').

In a second experiment, we tested the script against a cipher of another type, the *Borg* cipher,[4] based on simple substitution with a small nomenclature, encoded by 34 graphical signs with word boundaries marked by space. Since this cipher is based on simple substitution, rather than homophonic substitution, and with word boundaries already marked, it is less ambiguous than the Francia cipher. Accordingly, approximately 97% of the

[4] https://cl.lingfil.uu.se/~bea/borg/

words in the output from the cipher-key mapping script are identical to the words in the gold standard. The mismatches are mainly due to some word-initial upper-case letters in the ciphertext being written as the plaintext letter, instead of being encoded. As an example, the Latin word *nucem* (inflectional form of the word 'nut') would normally be enciphered as '9diw1' in the Borg cipher, but in one case it occurs instead as 'Ndiw1'. There are several similar cases for other words throughout the cipher.

4.2 Language identification

For the language identification task, we can see from Table 2 that both the Barb cipher and the Francia cipher are correctly identified as written in Italian by the CKM algorithm, and the Copiale cipher is correctly identified as written in German. These guesses are based on the fact that 79.17% of the tokens in the automatically recovered version of the Barb plaintext, and 80.05% of the tokens in the Francia text, could also be found in the Italian language model, whereas 86.55% of the tokens in the Copiale cipher could be matched in the German language model. As could be expected, the second best guess produced by the algorithm for the Italian manuscripts is for the closely related languages Spanish and Portuguese respectively. More surprisingly, the third best guess for both these texts is German, with a substantial amount of the tokens found in the German language model as well. A closer look at the German language model used in our experiments reveals a possible explanation to this. The German language model is based on data from the time period 1050–1914, where the oldest texts contain a substantial amount of citations and text blocks actually written in Latin, a language closely related to Italian. This might also explain why the Borg text, written in Latin, is identified by the script as written in German. The third guess for the Borg text is Swedish, for which the language model is also based on very old text (from 1350 and onwards), with blocks of Latin text in it. The Latin language model on the other hand is rather small, containing only about 79,000 tokens extracted from the *Ancient Greek and Latin Dependency Treebank*.[5] Due to the small size of this language model as compared to the language models for the other languages in this study, in combination with the fact that Latin words occur in older texts for many languages, it is hard for the script to correctly identify Latin as the source language.

For the German Copiale cipher, the second best guess is for Slovene, and the third best guess is for Swedish. This could be due to the fact that both Slovene and Swedish were strongly influenced by the German language in historical times, meaning that many German and German-like words would appear in historical Slovene and Swedish texts.

4.2.1 Present-day language models

So far, we have presumed that language models based on historical text would be best suited for the task of language identification in the context of historical cryptology. This is based on the assumption that spelling and vocabulary were different in historical times than in present-day text, meaning that some words and their particular spelling variants would only occur in historical text. It could however be argued that it is easier to find large amounts of present-day text to build language models from. As a small test to indicate whether or not present-day text would be useful in this context, we downloaded the Spacy language models for present day Italian[6] and German,[7] trained on Wikipedia text, and compared the coverage in these language models to the coverage in the historical language models, when applied to the plaintexts of the Barb, Francia and Copiale manuscripts.

As seen from Table 3, the percentage of word forms found in the language models based on present-day data is considerably lower than for the historical language models, even though the present-day data sets are larger. The preliminary conclusion is that language models based on historical text is better suited for the task at hand, but present-day language models could also be useful, in particular in cases where it is hard to find suitable historical data to train a language model. More thorough experiments would however be needed to confirm this.

[5] https://perseusdl.github.io/treebank_data/

[6] https://github.com/explosion/spacy-models/releases//tag/it_core_news_sm-2.1.0

[7] https://github.com/explosion/spacy-models/releases//tag/de_core_news_md-2.1.0

Name	Top 3 language models		Gold language
Barb-6956	**Italian**	**79.17%**	Italian
	Spanish	66.77%	
	German	65.73%	
Francia-64	**Italian**	**80.05%**	Italian
	Portuguese	72.40%	
	German	70.22%	
Borg.lat.898	German	56.73%	Latin
	Spanish	55.01%	
	Swedish	51.50%	
Copiale	**German**	86.55%	German
	Slovene	69.95%	
	Swedish	57.76%	

Table 2: Language identification results.

Name	Historical LM	Present-day LM
Barb-6956	79.17%	64.11%
Francia-64	80.05%	66.23%
Copiale	86.55%	81.44%

Table 3: Language identification results, comparing language models based on historical text to language models based on present-day text.

5 Discussion

From our experiments, we can conclude that the implemented algorithm makes it possible to restore the hidden plaintext from ciphertexts and their corresponding key. For one of the ciphers evaluated, 79% of the words were correctly mapped to the gold standard plaintext words, and the mismatches were mainly due to diacritics and repeated characters not being part of the key. This knowledge could easily be taken into consideration in further development of the algorithm, where the script could test to add diacritics in strategic positions and to repeat certain characters in cases where a specific word could not be found in a language model (provided that we already have an educated guess on what language the underlying plaintext is written in). For the other cipher evaluated, being an out-of-domain manuscript of a different cipher type and another underlying language than the manuscript used during training, we got very encouraging results with 97% of the words in the manuscript being correctly matched, and the mismatches in the remaining words mainly being due to plaintext characters occurring as part of ciphertext words.

For the language identification task, the results are mixed. The German and Italian manuscripts are correctly identified as being written in German and Italian respectively, whereas the algorithm assumes the Latin text to be written in German. This indicates that we need to be careful about what texts to put into the language models. In the current experiments, we have simply used the language models at hand for historical texts for different languages, without taking into account differences in time periods and genres covered, nor the size of the text material used as a basis for the language model. Thus, the language models used in the experiments are very different in size, where the Latin language model contains about 79,000 tokens, as compared to approximately 124 million tokens in the German language model. Furthermore, since the language in very old texts is typically quite different from the language in younger texts, language models only containing texts from the time period in which the cipher is assumed to have been created would better suit our purposes. In addition, many old texts contain blocks of Latin words, since this was the Lingua Franca in large parts of the world in historical times. This results in many Latin words being found in language models for other languages as well.

The language detection evaluation also shows that using language models based on historical text has a clear advantage over using state-of-the-art

language models based on present-day language.

6 Conclusion

In this paper, we have presented a study within the field of historical cryptology, an area strongly related to digital humanities in general, and digital philology in particular. More specifically, we have introduced an algorithm for automatically mapping encrypted ciphertext sequences to their corresponding key, in order to reconstruct the plaintext describing the underlying message of the cipher. Since ciphertexts and their corresponding keys are often stored in separate archives around the world, without knowledge about which key belongs to which ciphertext, such an algorithm could help in connecting ciphertexts to their corresponding keys, revealing the enciphered information to historians and other researchers with an interest in historical sources.

Acknowledgments

This work has been supported by the Swedish Research Council, grant 2018-06074: DECRYPT - Decryption of historical manuscripts.

References

Belgium Algemeen Rijksarchief, Brussels. 1647-1698. cl.lingfil.uu.se/decode/database/record/960 [link].

Craig Bauer. 2017. *Unsolved! The History and Mystery of the World's Greatest Ciphers from Ancient Egypt to Online Secret Societies*. Princeton University Press, Princeton, USA.

Friedrich Bauer. 2007. *Decrypted Secrets — Methods and Maxims of Cryptology*. 4th edition. Springer.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. IRSTLM: an open source toolkit for handling large scale language models. In *Proceedings of Interspeech 2008*, pages 1618–1621.

David Kahn. 1996. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. New York.

Kevin Knight, Beáta Megyesi, and Christiane Schaefer. 2011. The copiale cipher. In *Invited talk at ACL Workshop on Building and Using Comparable Corpora (BUCC)*. Association for Computational Linguistics.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. https://www.aclweb.org/anthology/P06-2065 Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 499–506, Sydney, Australia. Association for Computational Linguistics.

Benedek Láng. 2018. *Real Life Cryptology: Ciphers and Secrets in Early Modern Hungary*. Atlantis Press, Amsterdam University Press.

Beáta Megyesi, Nils Blomqvist, and Eva Pettersson. 2019. The DECODE Database: Collection of Ciphers and Keys. In *Proceedings of the 2nd International Conference on Historical Cryptology, HistoCrypt19*, Mons, Belgium.

Malte Nuhn and Kevin Knight. 2014. Cipher type detection. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1769–1773. Association for Computational Linguistics.

Eva Pettersson and Beáta Megyesi. 2018. The HistCorp Collection of Historical Corpora and Resources. In *Proceedings of the Digital Humanities in the Nordic Countries 3rd Conference*, Helsinki, Finland.

Sujith Ravi and Kevin Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 812–819. Association for Computational Linguistics.

Simon Singh. 2000. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchor Books.