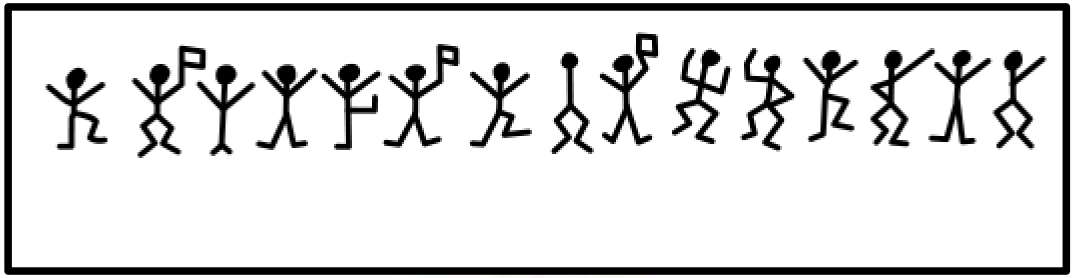
Computational Decipherment of Unknown Scripts

Bradley Hauer
Department of Computing Science

UNIVERSITY OF ALBERTA

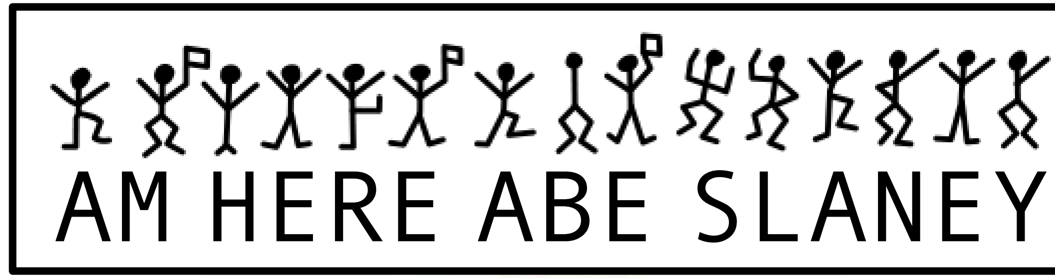
Introduction

- "VE AYO Y FGVUQE INXK KYC VP YMGVX..."
- These texts have been *enciphered*



Introduction

- "VE AYO Y FGVUQE INXK KYC VP YMGVX..."
- These texts have been *enciphered*
- *Decipherment*: Recover the original *plaintext* for a given *ciphertext*

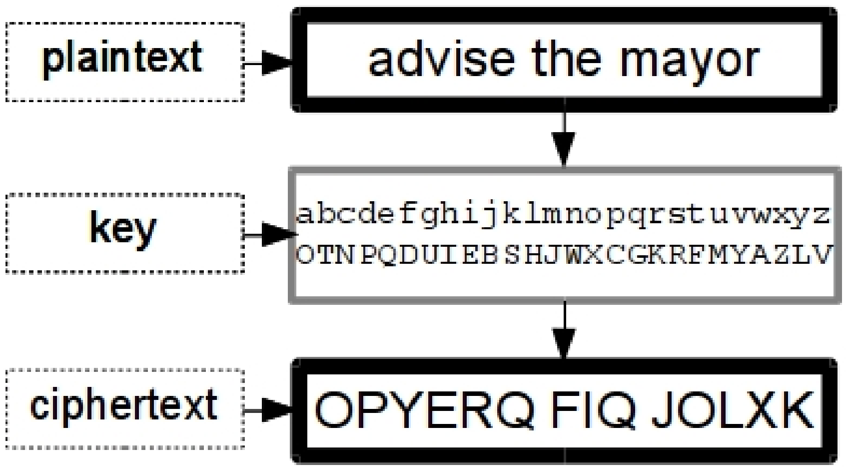


Overview

- Task definition and prior work
- New method for solving substitution ciphers
- Decoding texts in unknown language & script
  - Ciphertext language identification
  - Anagram decipherment
  - Application to the Voynich manuscript

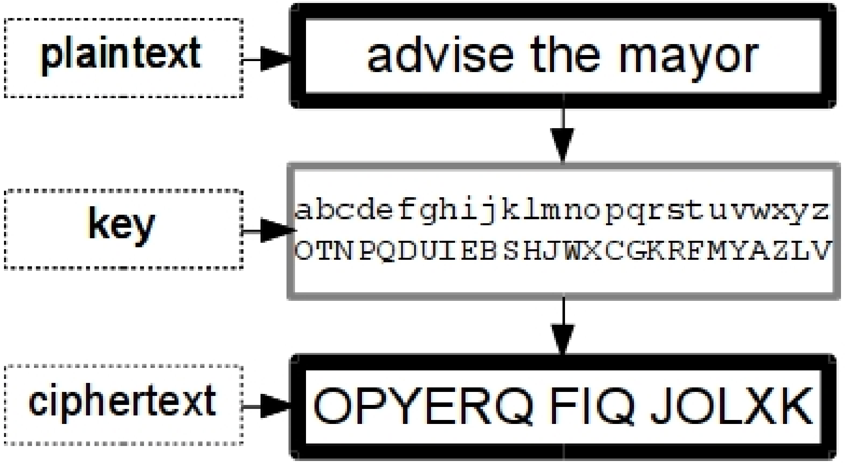
Decipherment: task definition

- *Monoalphabetic substitution ciphers*
- Key: Each letter is mapped uniquely to one cipher symbol



Decipherment: task definition

- Given a ciphertext, and a corpus of text (millions of words) in the plaintext language, find the plaintext
  - Find the right key
- Performance: decipherment character accuracy



Low-Order Character LMs

- Ravi and Knight (2008, EMNLP)
  - Low-order character language models
  - Integer programming, optimal solution
- `"VE AYO Y FGVUQE INXK KYC VP YMGVX"`

  `"ae cor o blathe wind dof as oulan"`

Dictionary Attack

- Olson (2007, Cryptologia)
  - Dictionary attack
  - Use a dictionary to select possible words
- `"VE AYO Y FGVUQE INXK KYC VP YMGVX"`

  `"us far a youngs with had up about"`

Hill Climbing with Random Restarts

- Norvig (2009, from *Beautiful Data*)
  - Hill-climbing with a character language model to generate candidate solutions
  - Word language model to choose a solution
  - Does not assume spaces are preserved
- `"VEAY OYF GVU QEINXKKY CVPYMGV X"`

  `"ache red tab scoville magenta i"`

Higher-Order Character LMs

- Nuhn et al. (2013, ACL)
  - Higher order (up to 6-gram) character LMs
    - Possible due to heuristic search (Beam Search)
  - State of the art (before my project)
- `"VE AYO Y FGVUQE INXK KYC VP YMGVX"`

  `"in pay a utilon mesh had if artis"`

Our Method

"VE AYO Y FGVUQE INXK KYC VP YMGVX"

Our Method

"VE AYO Y FGVUQE INXK KYC VP YMGVX"

"it was a bright cold day in april"

(opening line from George Orwell's *Nineteen Eighty-Four*)

Our Method

`"VE AYO Y FGVUQE INXK KYC VP YMGVX"`

`"it was a bright cold day in april"`

(opening line from George Orwell's *Nineteen Eighty-Four*)

- Key mutation – modify a key to find new keys
- Key scoring – evaluate and compare keys
- Tree search – search for a good key

Key Mutation

- Given a guess of what the correct key is, come up with one or more new guesses
  - New keys from old keys
- Word *n*-grams are pattern-equivalent (or *p*-equivalent) if there exists a monoalphabetic substitution that transforms one into the other
  - 'will' is *p*-equivalent to 'jazz'
  - 'will' is <u>not</u> *p*-equivalent to 'said'
  - 'am I not' is *p*-equivalent to 'in a red'

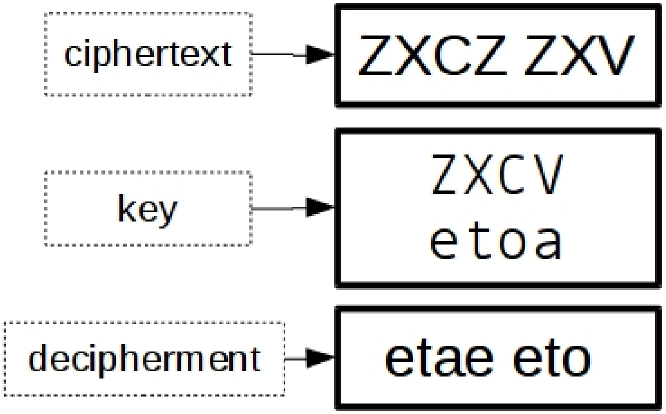
Key Mutation

- Take a ciphertext word unigram, bigram, or trigram, call it *X*
- Choose a *p*-equivalent *n*-gram from the plaintext language, call it *Y*
- Modify the key such that *X* deciphers to *Y*

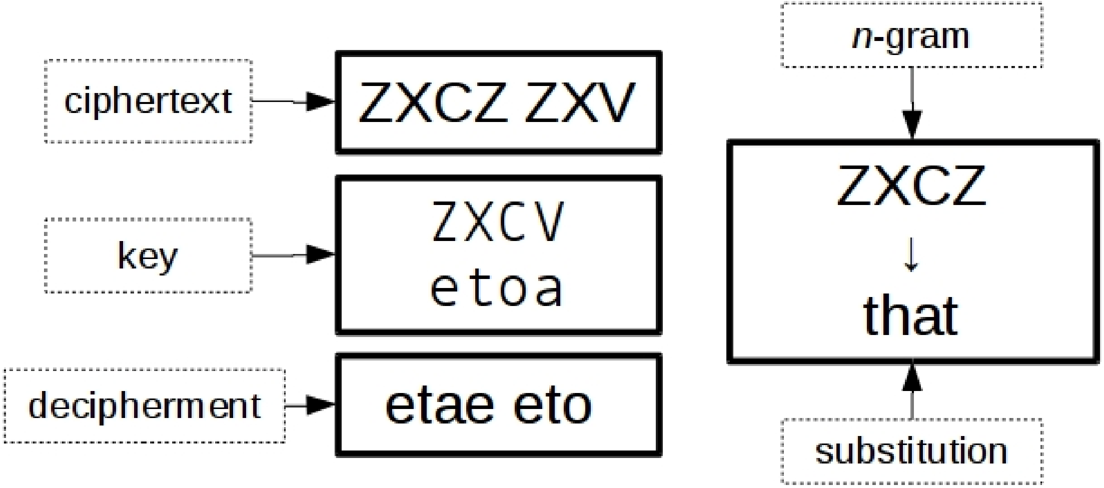
Key Mutation

- Take a ciphertext word unigram, bigram, or trigram, call it *X*
- Choose a *p*-equivalent *n*-gram from the plaintext language, call it *Y*
- Modify the key such that *X* deciphers to *Y*
- Repeat for all choices of *X*, many choices of *Y*, to generate a (large) set of new keys

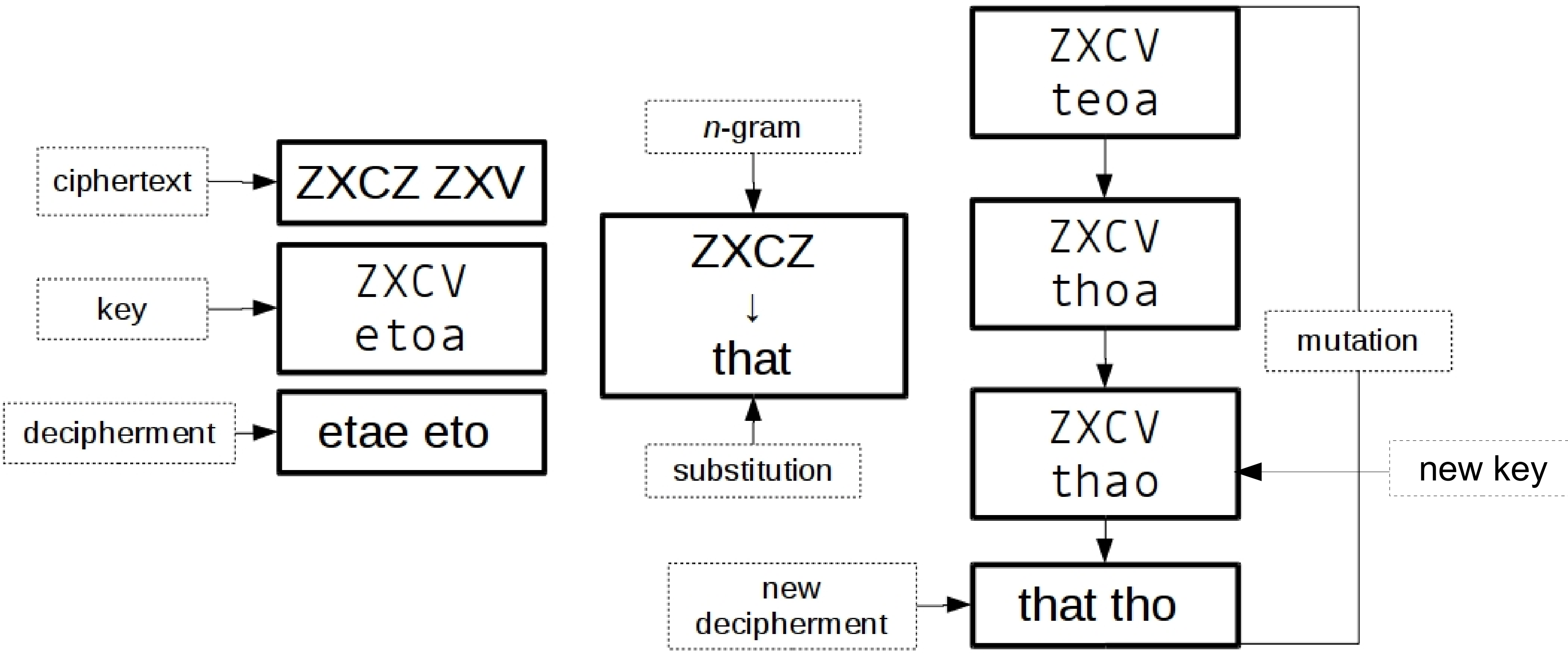
Key Mutation - Example



Key Mutation - Example



Key Mutation - Example



Key Scoring

- A key induces a decipherment of the ciphertext
- The decipherment should "look like" the plaintext language
- *N*-gram language models
  - Derived from training corpus
- Combine language models:
  - Context size: unigram, bigram, trigram
  - Unit of analysis: character, word

Key Scoring – Example

- Consider two possible decipherments of our test cipher

```
bc tan a public from may bs adubo
it may a flight crew was in ablie
```

- Neither looks correct, but to continue our search, we need to decide which is better

Key Scoring – Word LM

(1) `bc tan a public from may bs adubo`

(2) has more common unigrams and a more common bigram

(2) `it may a flight crew was in ablie`

Key Scoring – Word LM

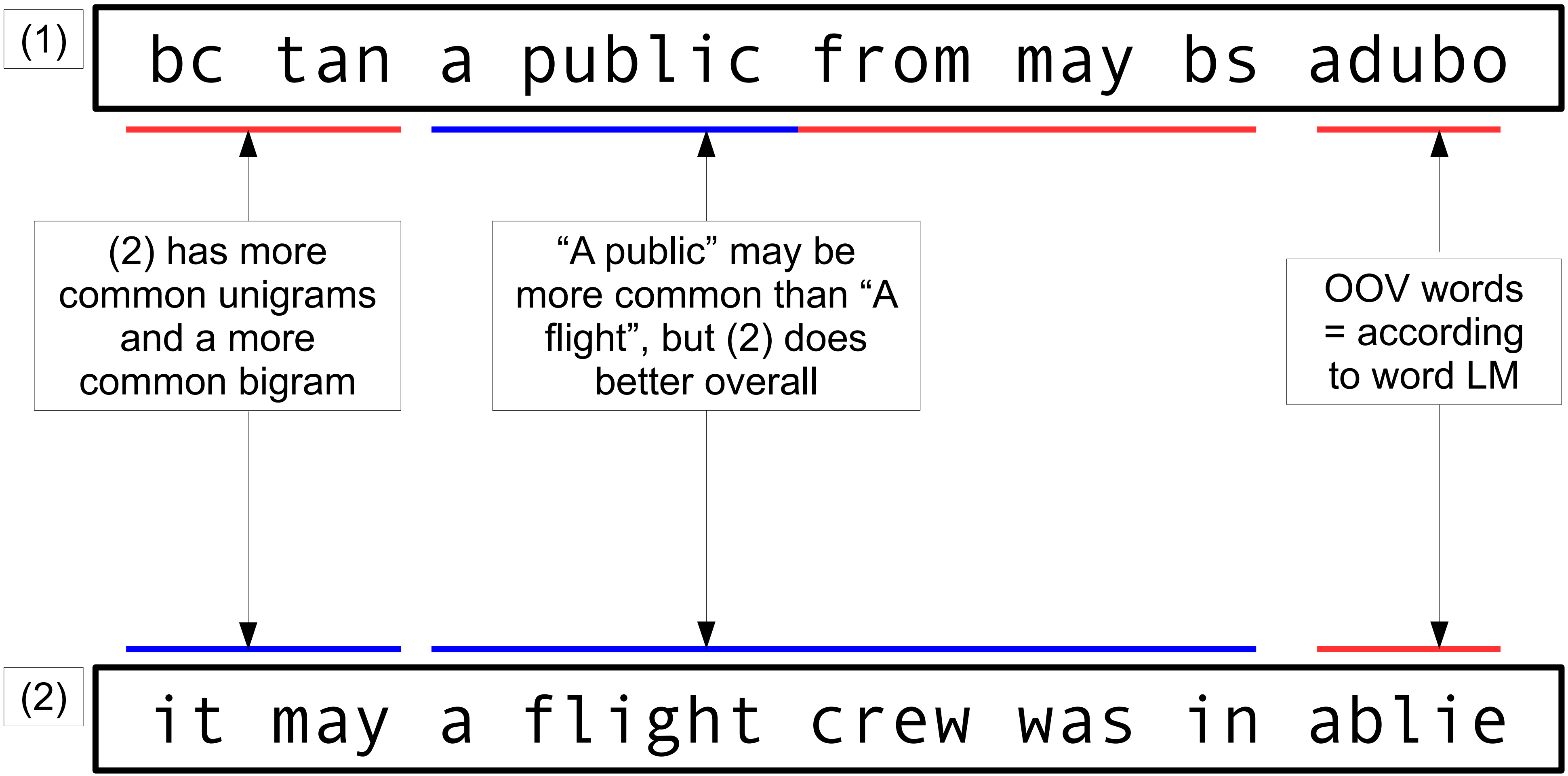
(1) `bc tan a public from may bs adubo`

(2) has more common unigrams and a more common bigram

"A public" may be more common than "A flight", but (2) does better overall

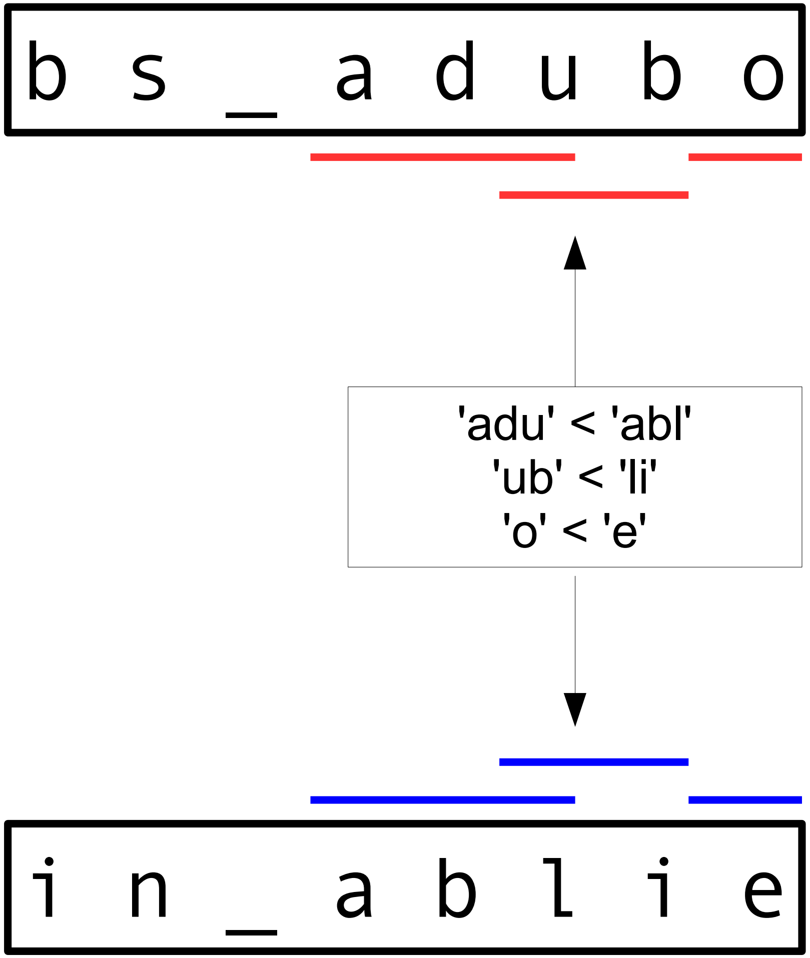
(2) `it may a flight crew was in ablie`

Key Scoring – Word LM



Key Scoring – Character LM

- Can apply the same idea at the character level
  - Unigrams, bigrams, trigrams
- Apply to entire string
- Most important for OOV words



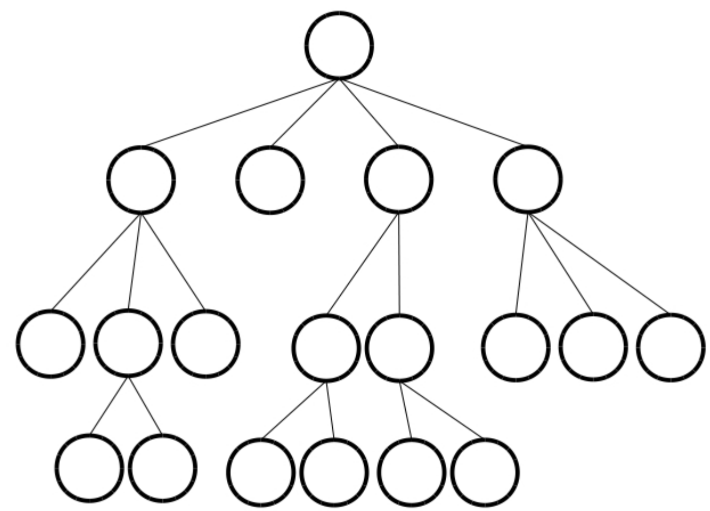
Key Scoring – Combination

- For words and characters separately: Combine unigram, bigram, and trigram probabilities through interpolation (weighted average)
- Combine word and character information: weighted average of word and character log-probabilities

```
bc tan a public from may bs adubo: -75.5
it may a flight crew was in ablie: -58.7
```

Tree Search

- Key mutation produces a decipherment tree
- Exponential growth
- How to search the tree?
  - Want a good key, in reasonable time



Solver Search Strategies

- Beam Search
  - Mutate only the most promising *B* nodes from each level of the tree
- Monte Carlo Tree Search
  - Mutate only one node at a time
  - Balance exploration with exploitation

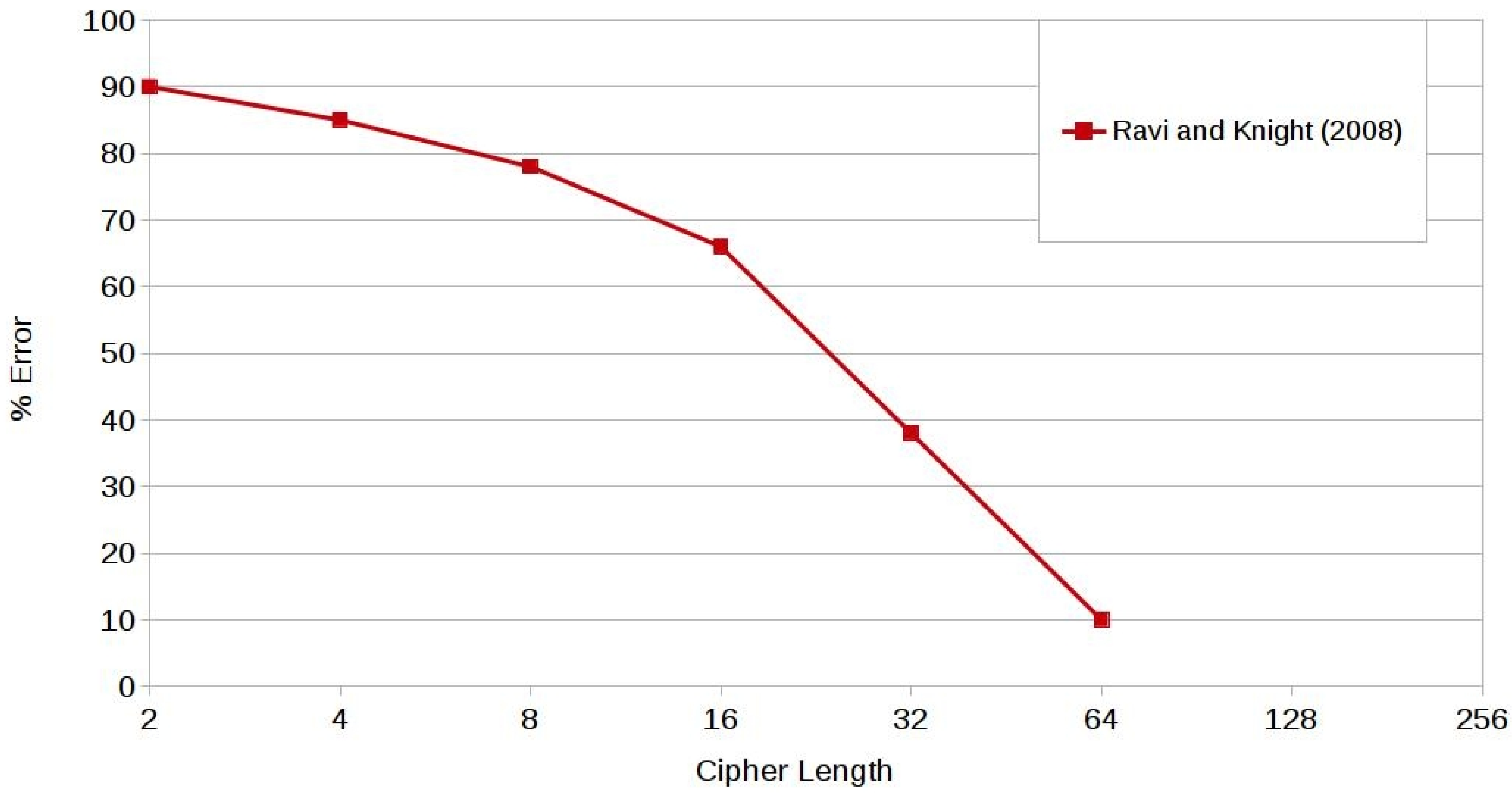
The New Method: Summary

- Build a key tree through key mutation
- Assign values to keys with key scoring
- Search the tree with Beam Search or MCTS
- Return the top scoring key

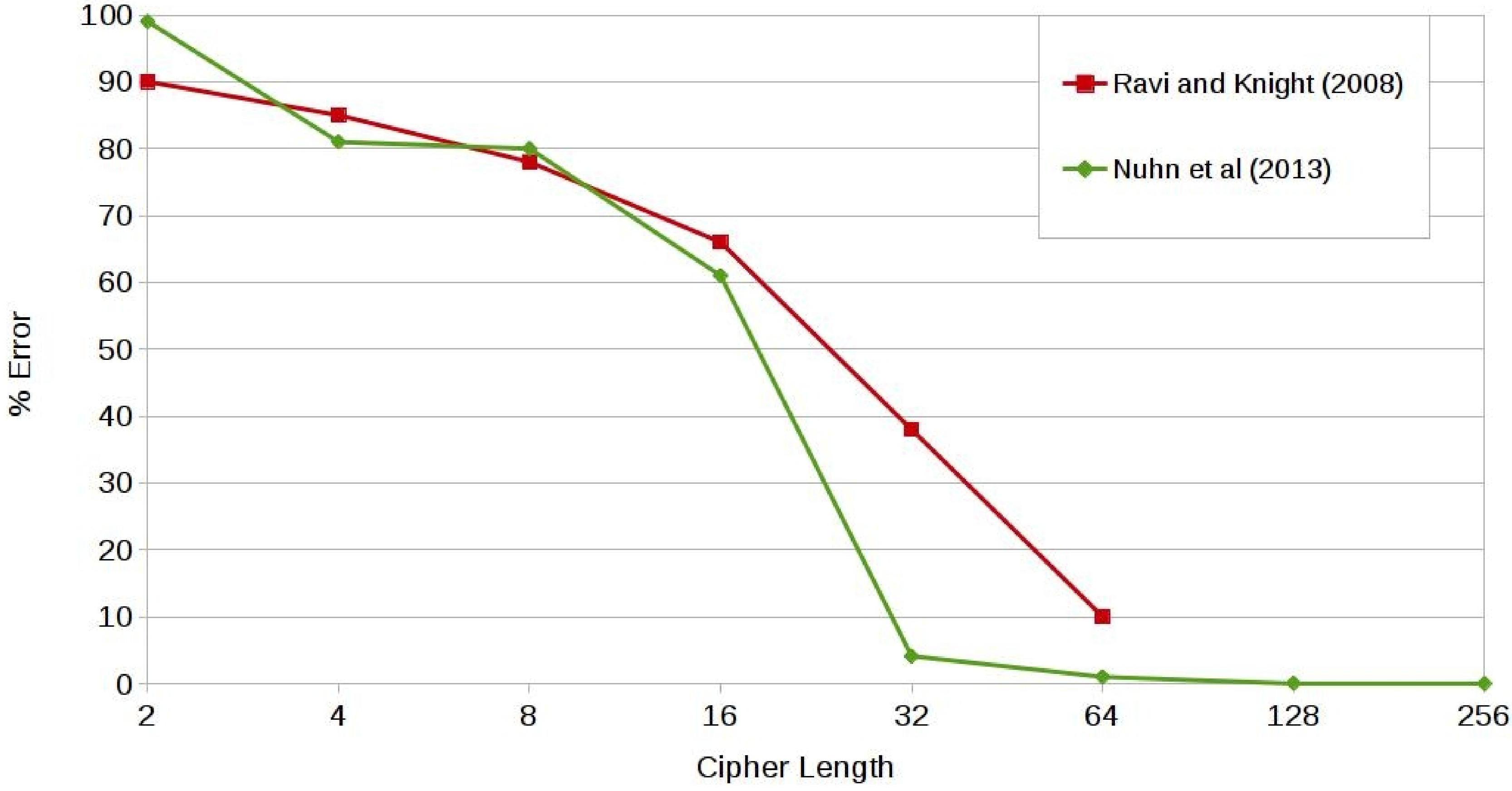
Experiments

- Training data: the New York Times
- 2 sets of test data: NYT (same domain), Wikipedia (different domain)
- For both test domains: 50 ciphers each of length 2, 4, 8, ..., 256, as in previous work

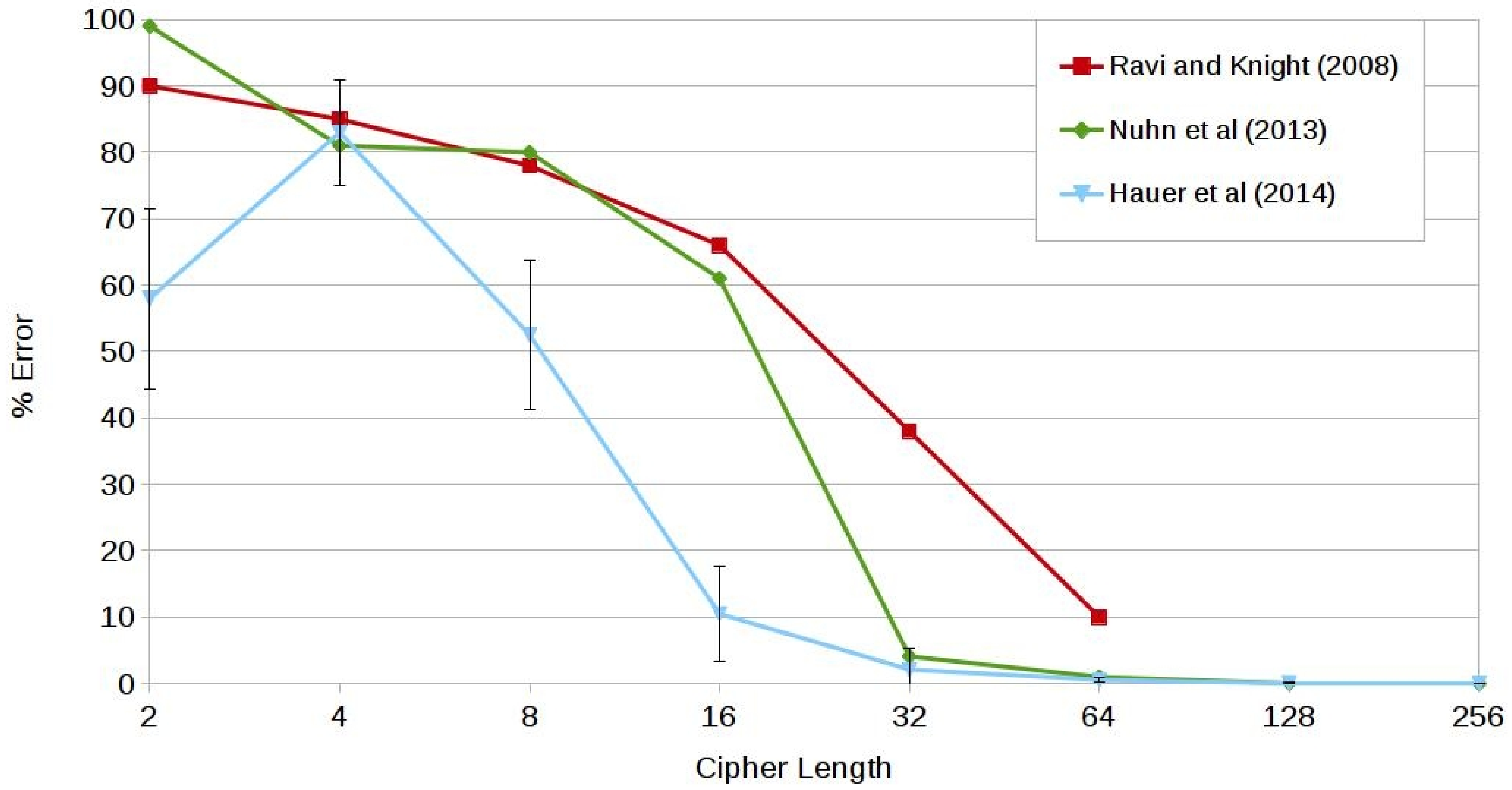
Results: Wikipedia Data



Results: Wikipedia Data



Results: Wikipedia Data



Results

- MCTS very close to Beam Search, much faster
- Same overall trends across both data sets

Applications

- With a simple modification, can solve ciphers without spaces
- Gold-Bug Cipher – Correctly solve a 204 character cipher without spaces from Edgar Alan Poe's *The Gold-Bug*
- Unsupervised transliteration – Decipher Cyrillic into Latin
- Deniable encryption – Create ciphertexts that look like plaintexts

Summary (So Far)

- New method of solving substitution ciphers
  - Combination of word and character LMs
  - P-equivalence-based key mutation function
  - Heuristic search
  - Outperforms prior work, state-of-the-art results

Overview

- Task definition and prior work
- New method for solving substitution ciphers
- Decoding texts in unknown language & script
  - Ciphertext language identification
  - Anagram decipherment
  - Application to the Voynich manuscript

An Open Decipherment Problem

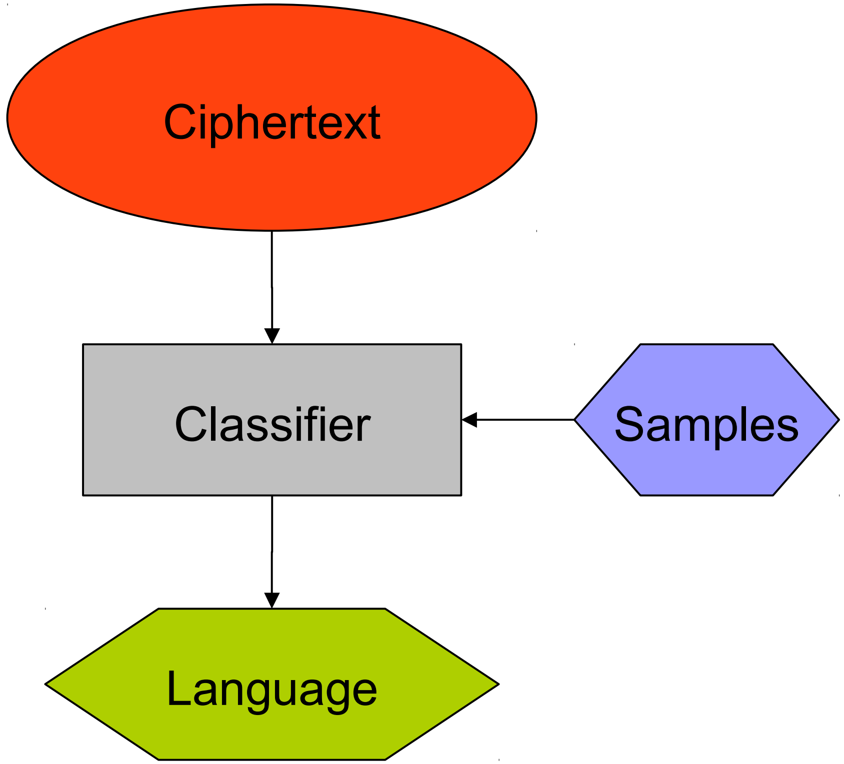
- The Voynich Manuscript (VMS)
- 15^{th} century
- Undeciphered
- Unique script
- Unknown language

Ciphertext Language Identification

- Identification of the underlying, or "ciphertext", language is important for decipherment:
  - Lost languages:
    - Egyptian hieroglyphics (Coptic), Linear B (Greek), Mayan glyphs (Ch'olti')
  - Ciphers:
    - Copiale Cipher (German; 18^{th} century, Knight et al. (2011))
  - Unknown encodings, optical character recognition...

Ciphertext Language Identification

- Have a ciphertext, plaintext language unknown
- Have short sample texts in many languages
  - ~1000 words
- Which is the language of the ciphertext?

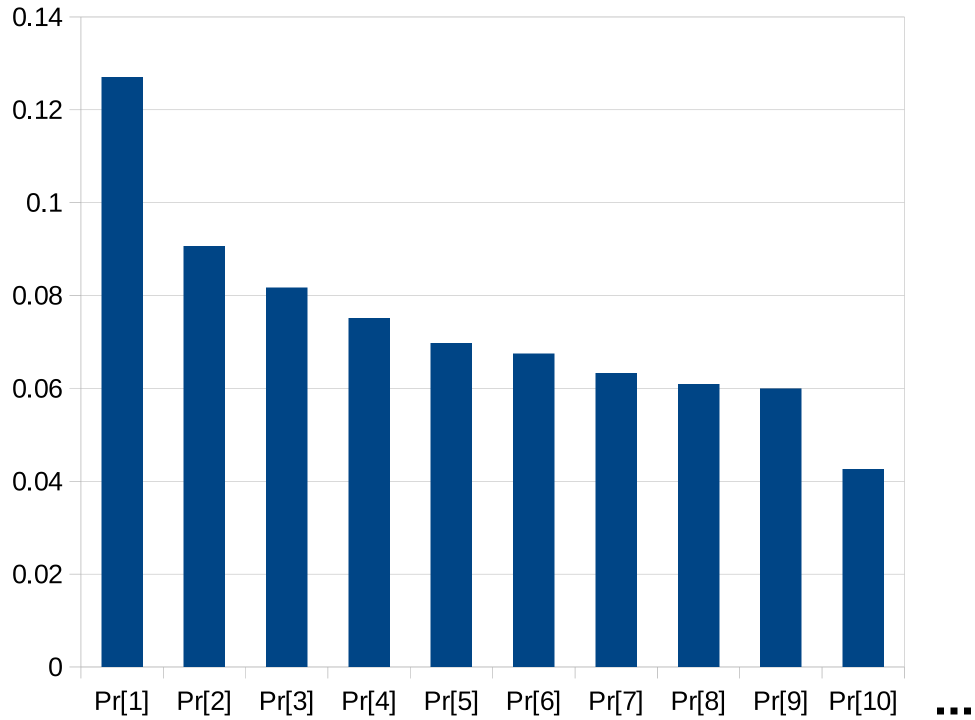


Methods

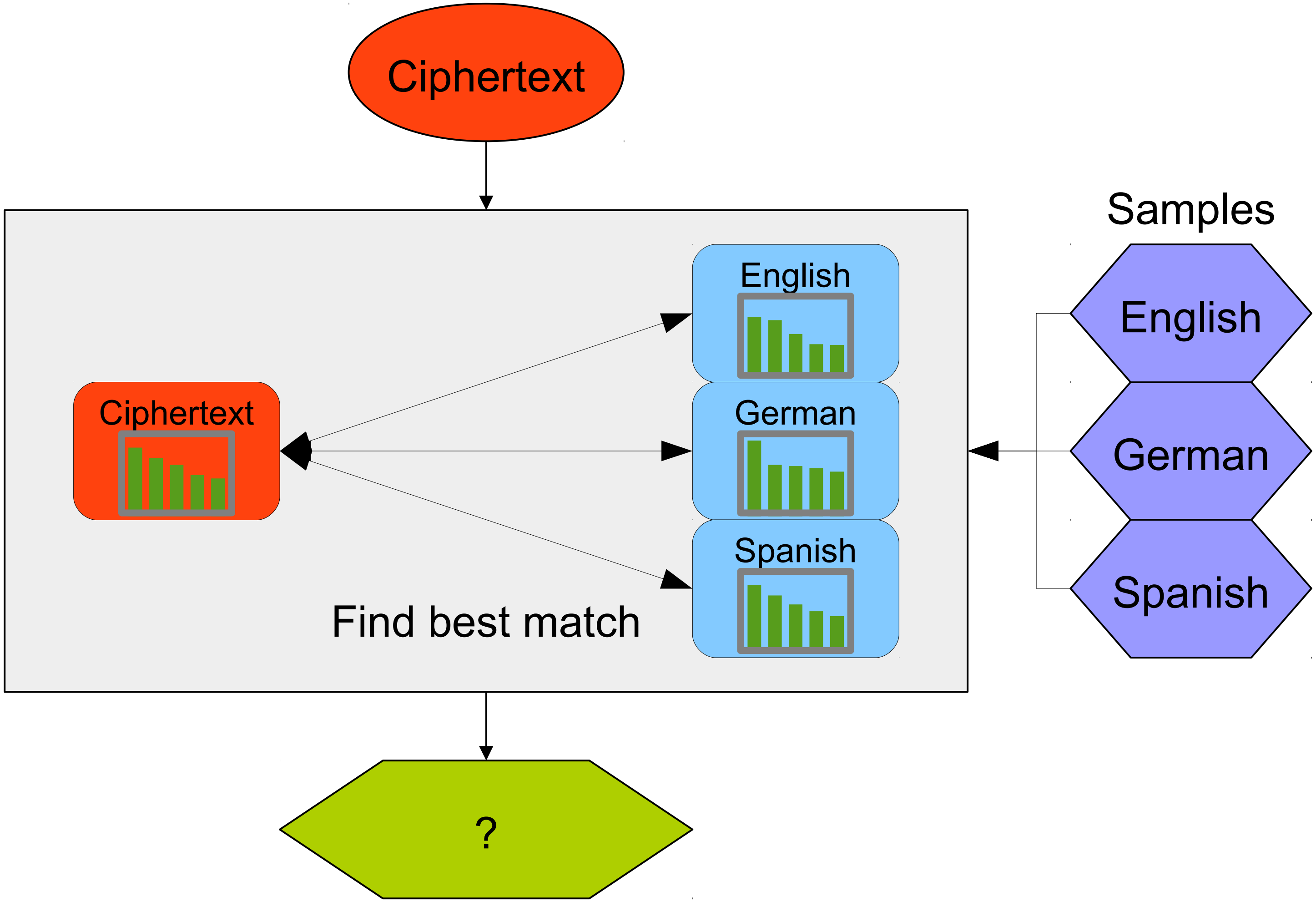
- Character frequency
- Decomposition pattern frequency
- Trial decipherment

Sorted Symbol Distribution

- Probability distribution: Pr[*i*] = probability that a random letter is the *i*th most frequent letter
  - Pr[1] = 13% for a typical English text, Pr[1] = 16% for a German one
- Resistant to encipherment!



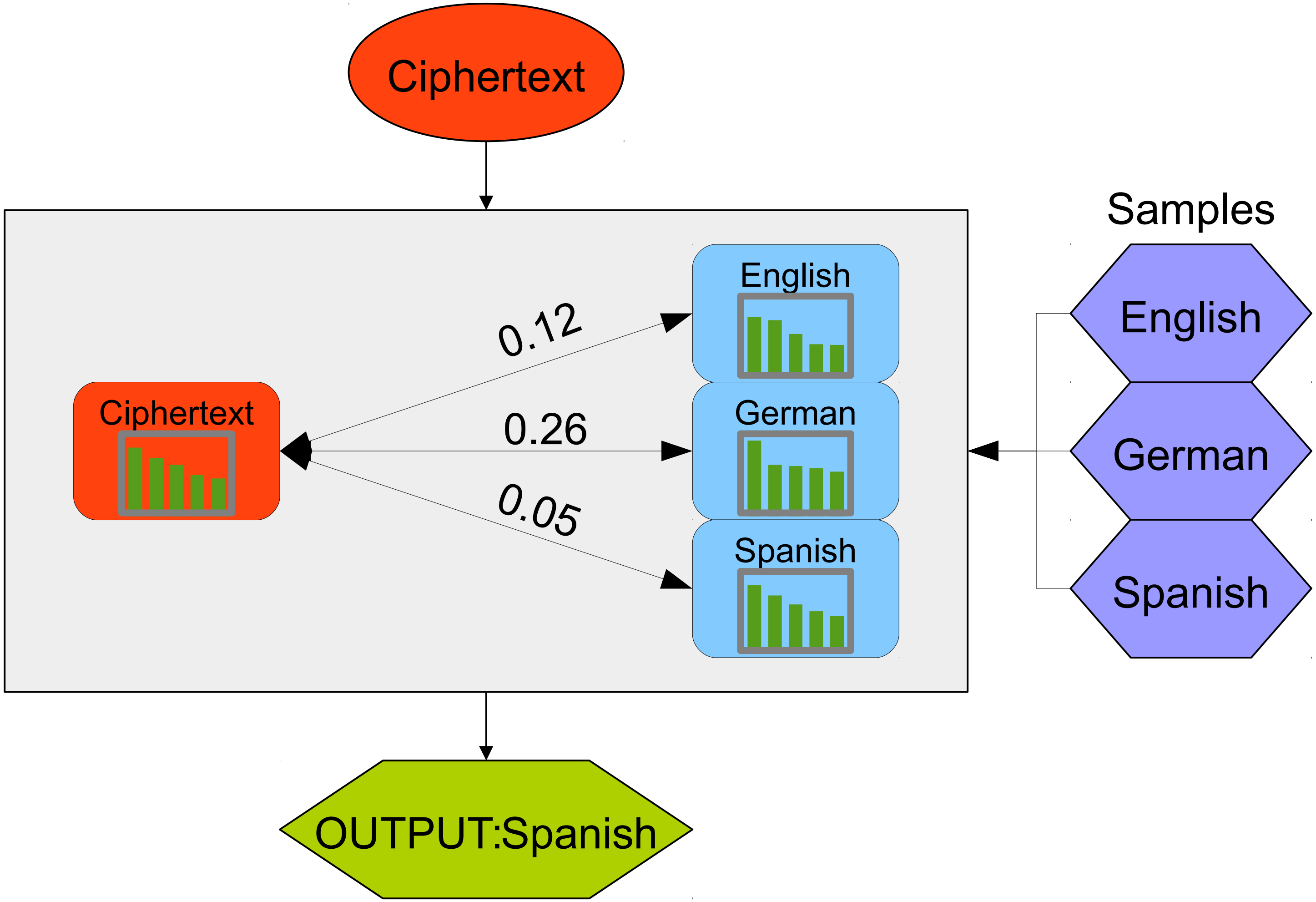
Classification



Bhattacharyya Distance

- Compare distributions *p, q*: $-\ln\left(\sum_x \sqrt{p(x)q(x)}\right)$
- We can measure the distance between the sorted symbol distributions of two texts
- Classify the ciphertext as the language of the sample with the nearest distribution

Classification



Method 2: Decomposition Patterns

- Idea: replace the sorted symbol distribution
  - Look at repeated letters within words
- "SEEMS" has two letters which occur twice, one which occurs once: pattern (2,2,1)
- "BEAMS" has five distinct letters: (1,1,1,1,1)
- "WERE": (2,1,1)
- Question: Do certain patterns occur more/less frequently in different languages?

Decomposition Patterns

- Answer: Yes!
- Patterns survive encipherment...
- ...so we can use them to distinguish different languages

Language	P[2,2,1]	P[1,1,1,1,1]
English	0.001878	0.056292
Bulgarian	0.000025	0.000473
German	0.005989	0.050476

Classification

- Construct probability distributions over patterns for the ciphertext and each language sample
  - What is the probability of a random word having a given pattern?
- Compute Bhattacharyya distance between ciphertext distribution and each sample distribution
- Classify ciphertext language as language of sample with nearest distribution

Method 3: Trial Decipherment

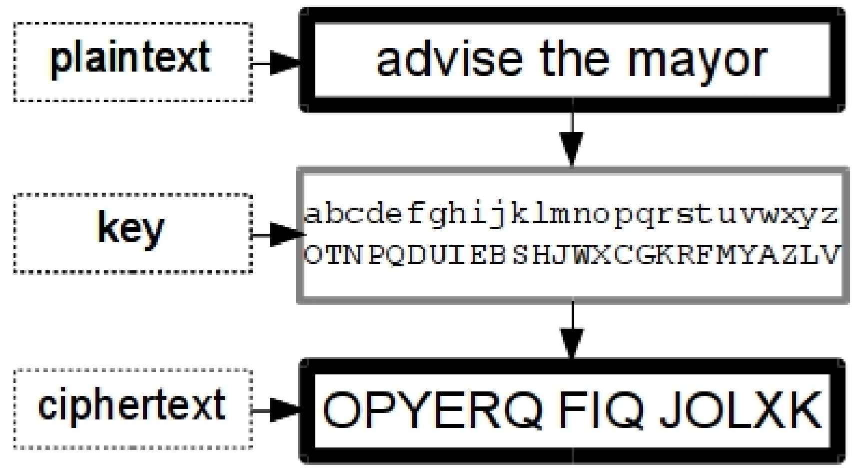
- Idea: Trying to decipher a ciphertext into the wrong language will almost certainly fail
  - e.g. deciphering a French ciphertext into English
- Try to decipher the ciphertext into each candidate language
- Whichever language gives the "best" decipherment is probably correct
  - "Best": most probable according to character LM

Example

- Decipher: "`zsvjxmnqo bpxww uz ylzz, xm wzxbm no mpz zwzgzomxld xos...`"
  - Italian: `edgsicuon alitt he pree, ic teiac un cle etemencirà ind...`
  - English: `education shall be free, at least in the elementary and...`
- The English decipherment is more probable
- So the ciphertext language is probably English

Decipherment

- Method must be fast
  - runs=languages*ciphers
- Work with small training data
  - Previous algorithm needs millions of words of training data!
- "Greedy Swap"



Greedy Swapping

- Consider all possible ways of swapping two letters in the key
- Choose the new key giving the best decipherment
- Repeat until no improvement

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
etaoinshrdlcumwfgypbvkjxqz
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
teaoinshrdlcumwfgypbvkjxqz
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
etaoizshrdlcumwfgypbvkjxqn
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
etloinshrdacumwfgypbvkjxqz
```

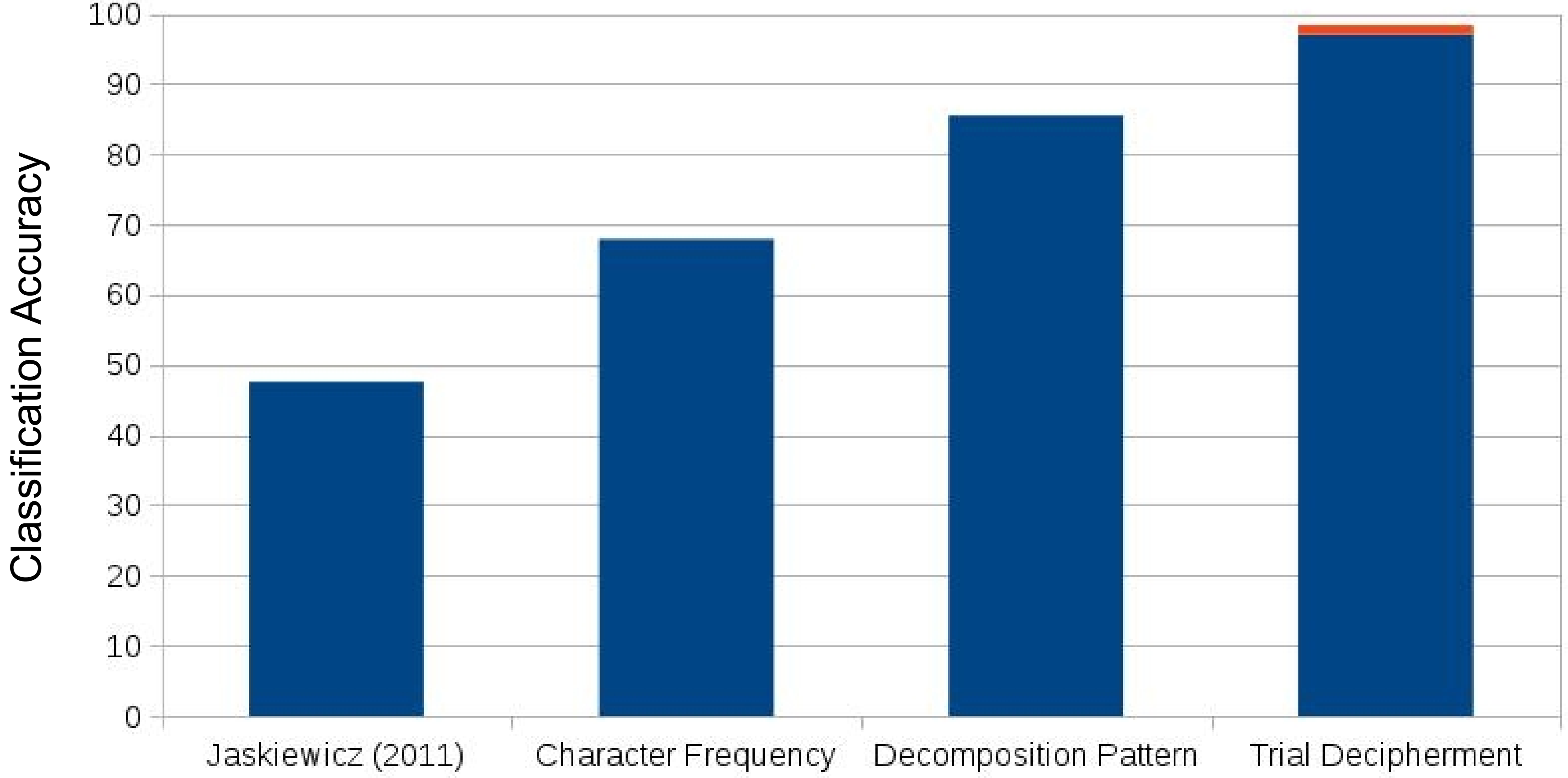
Classification

- We evaluate a decipherment by assigning it a probability with a bigram character language model
  - Minimal training data and time requirements
- Repeat for all languages
- Whichever language gives the best (most probable) decipherment is the language chosen

Experiments: Data

- Universal Declaration of Human Rights (UDHR)
  - Freely available, widely translated
  - Emerson et al. (2014): UDHR in 380 languages
  - 380 language samples, 380 ciphertexts
- Task: 380 languages, choose the ciphertext language
- Compare to Jaskiewicz (2011)

Results



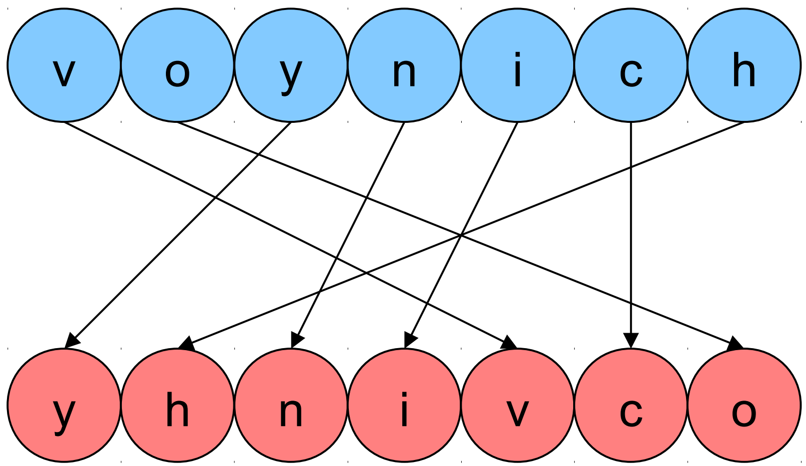
Classification Accuracy
100
90
80
70
60
50
40
30
20
10
0
Jaskiewicz (2011)
Character Frequency
Decomposition Pattern
Trial Decipherment

Discussion

- Good results by all three of our methods
- Different levels of assumptions:
  - Unigram assumes only a 1-1 cipher
  - Word patterns further assume knowledge of word boundaries
  - Decipherment uses a language model, which relies on letter order – letters must not be transposed

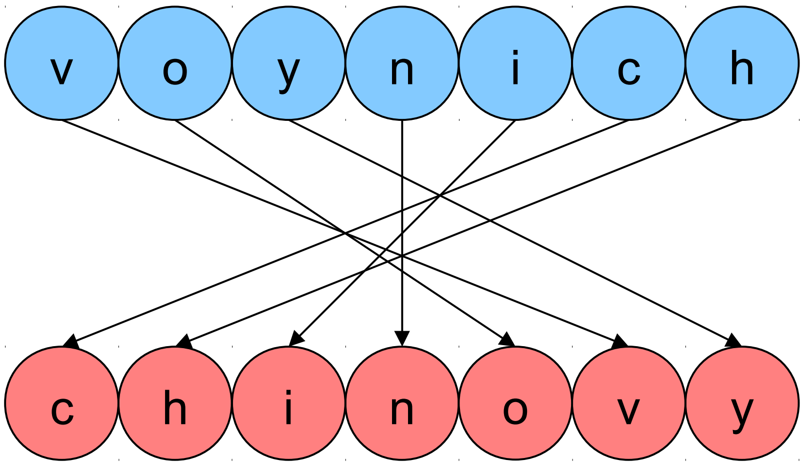
Anagramming

- Anagramming re-arranges letters within words
- Errors, writing direction, intentional...



Anagramming

- Anagramming re-arranges letters within words
- Errors, writing direction, intentional...
- Special case: alphagramming puts letters specifically into alphabetical order

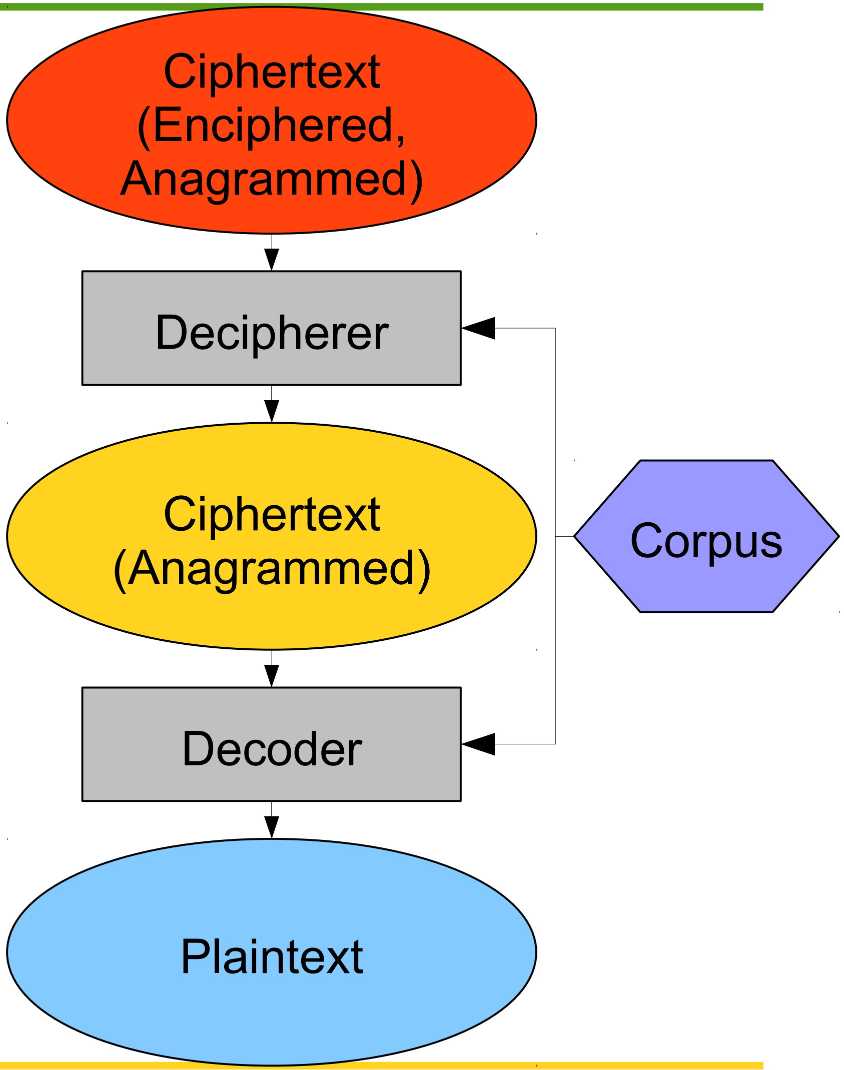


Anagramming and the Voynich manuscript (VMS)

- Rugg (2004): VMS text "superficially similar" to an alphagrammed text
- Reddy and Knight (2011) note unusually predictable character order
- Additional evidence discovered in our research

Anagram Decipherment

- 1-to-1 substitution cipher with anagramming
- Method: first reverse the substitution, then reverse the anagramming



Example

FGUV FV AN CKNRRYZ BS ACEGPQRRRUVZ

Decipher

hist is an aeelmpx fo cdeeehimnprt

Decode

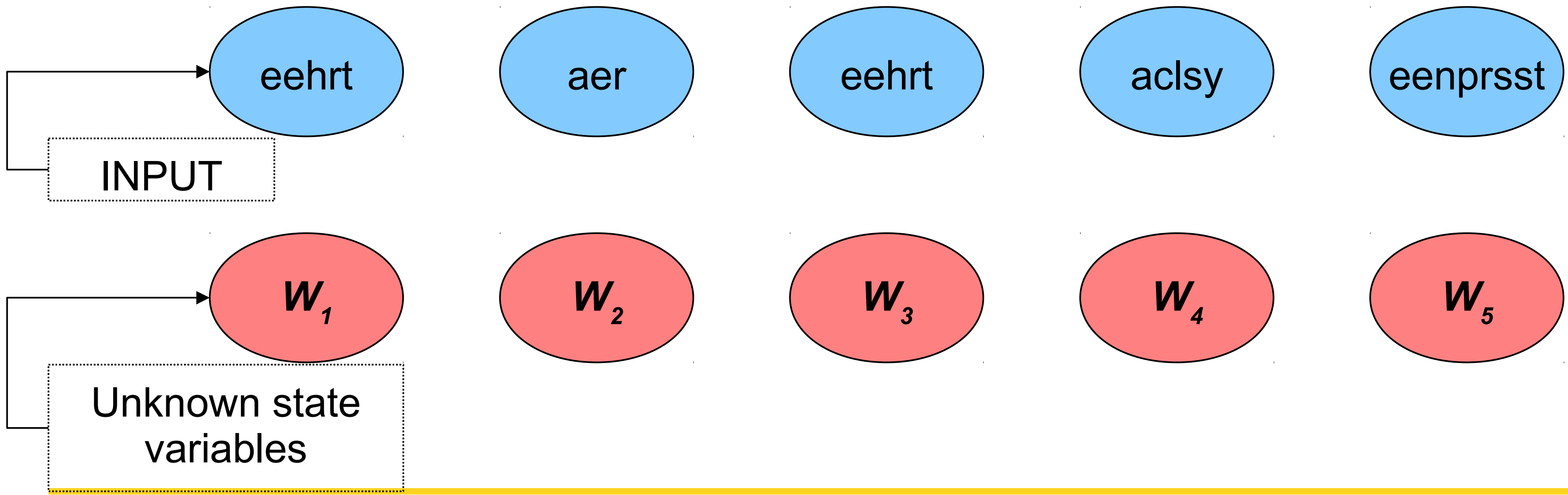
this is an example of decipherment

Decipherment

- The substitution solver presented earlier can be modified to work with anagrams
- Key point: we can undo the encipherment, reducing the problem to un-scrambling alphagrams

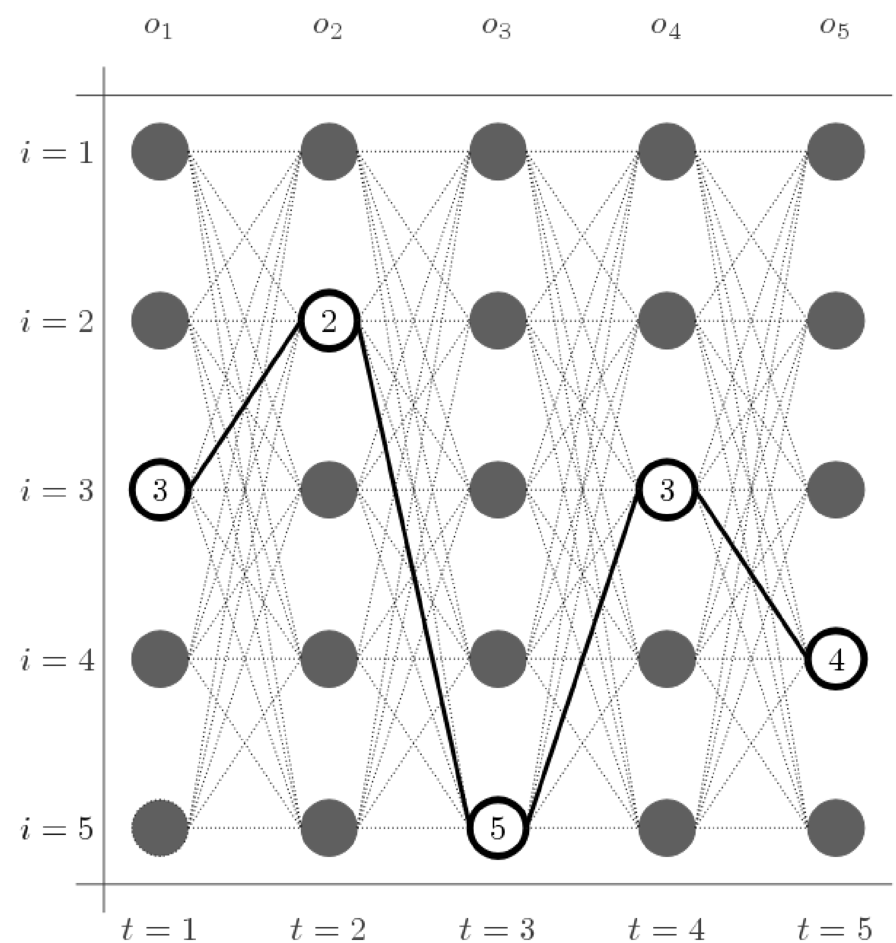
Decoding

- Can model the anagram decoding process with a Hidden Markov Model (HMM)
- Parallel sequences of **emissions** and **states**



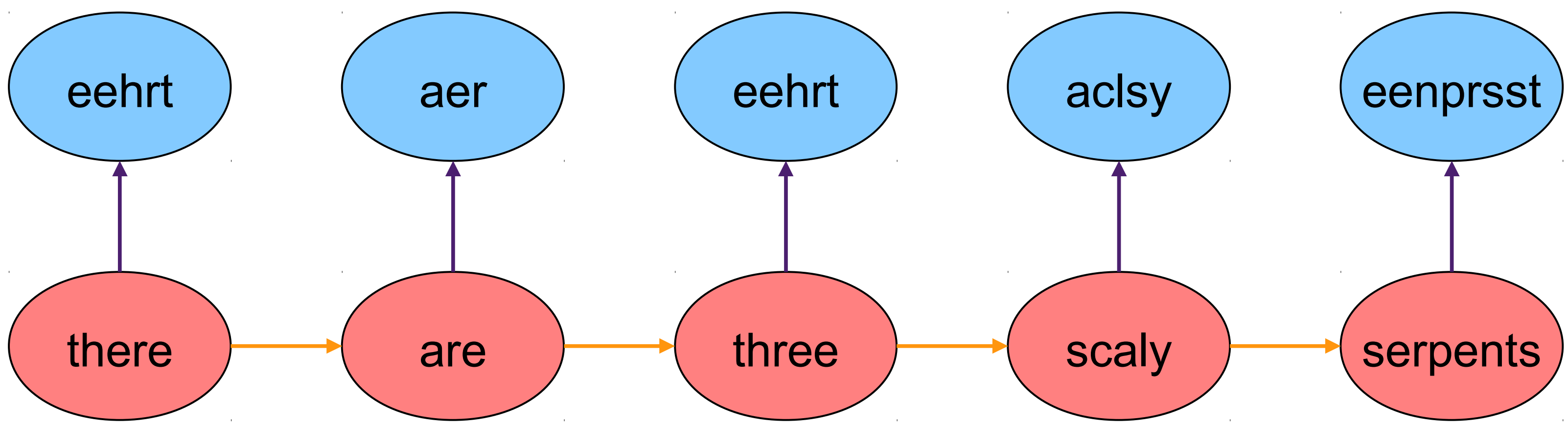
The Viterbi Algorithm

- Polynomial time decoding algorithm
- Guaranteed to find max. probability state sequence for a HMM



Viterbi Decoding

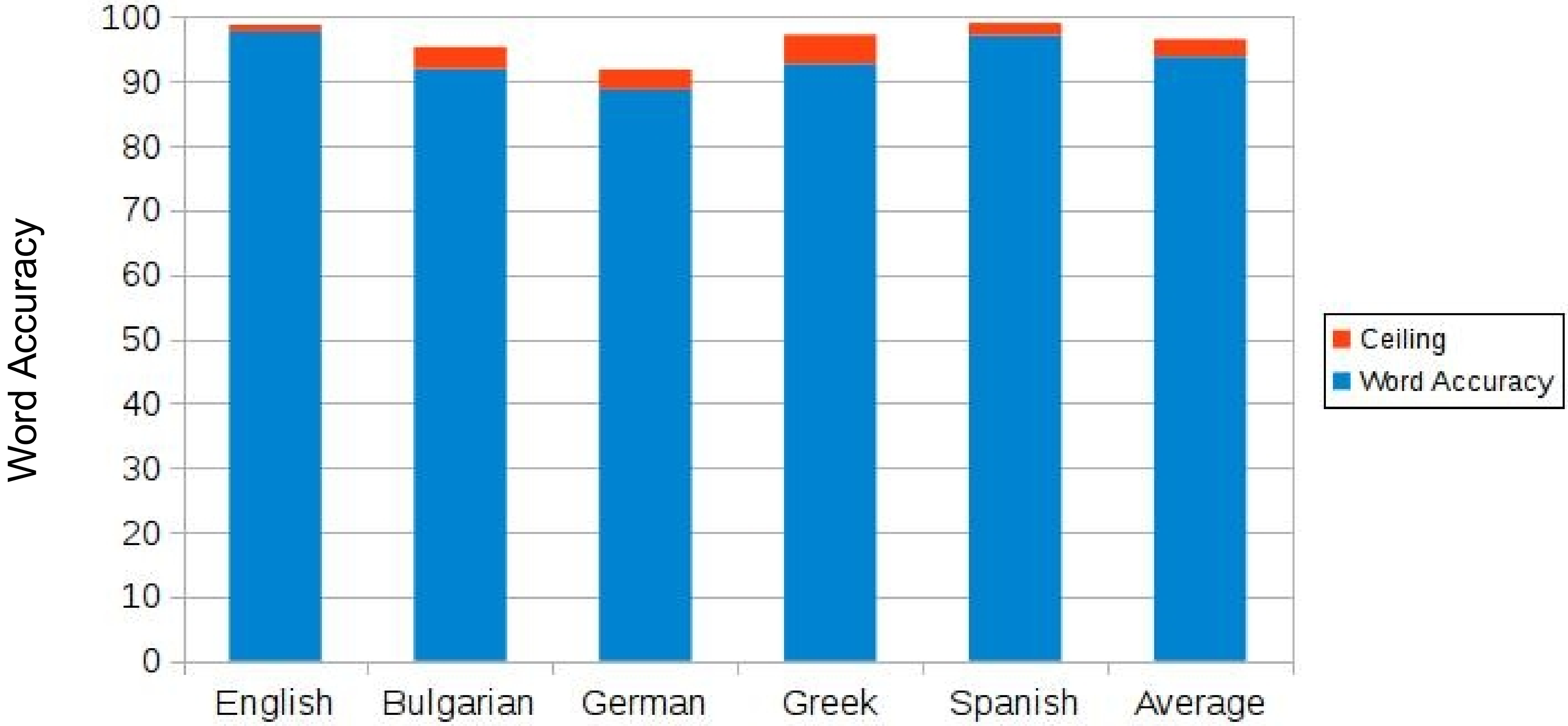
- Using just a dictionary and a trigram language model, the Viterbi Algorithm can efficiently compute the most probable word sequence!



Experiments

- Training data: Europarl corpora in Bulgarian, German, Greek, English, and Spanish
- Testing data: 50 Wikipedia articles (10 per language), anagrammed and enciphered
- Metric: word accuracy

Results – Word Accuracy



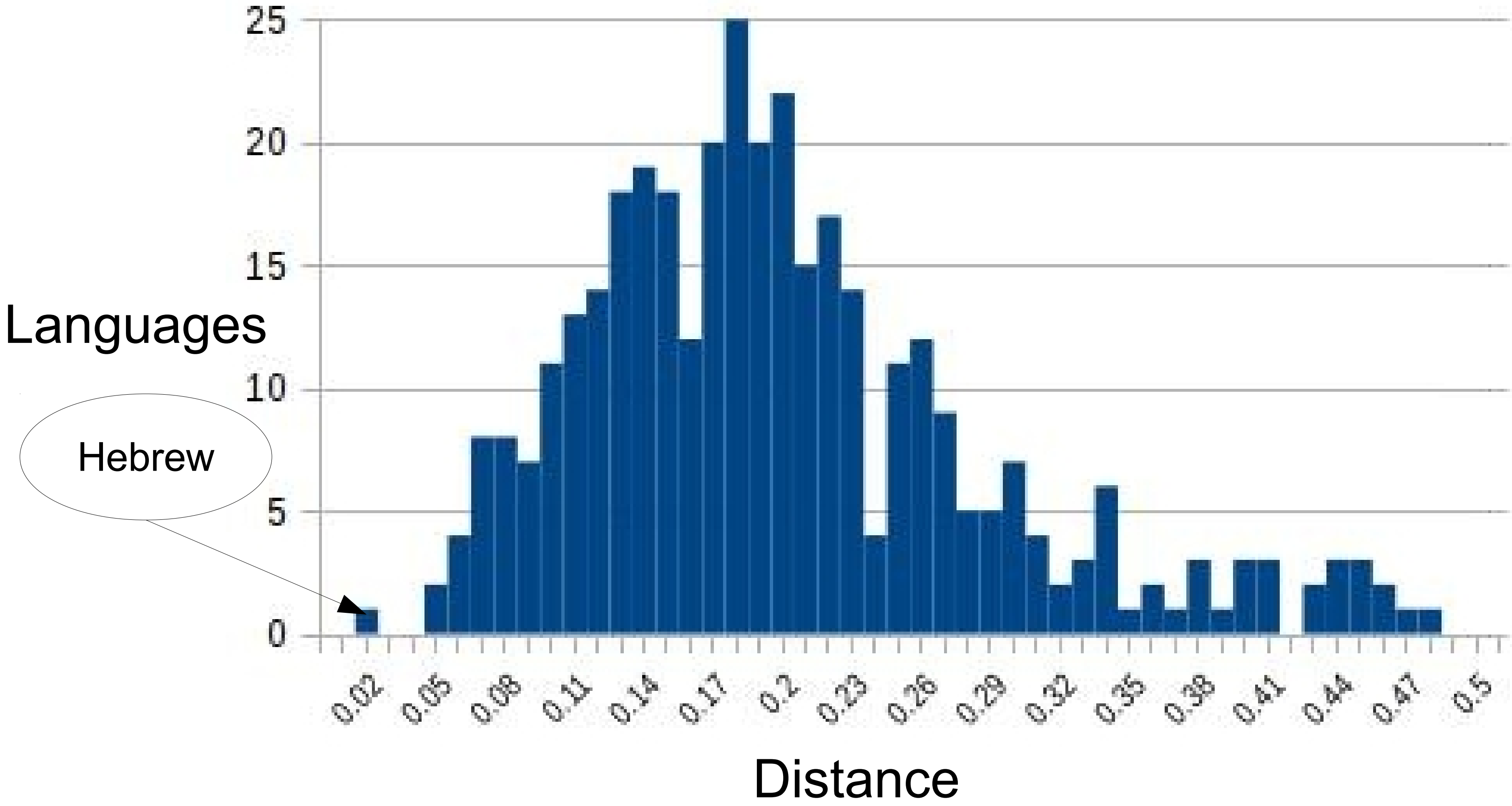
Voynich Experiments

- Methods in this part of the presentation were primarily motivated by the VMS
- Have seen that they are highly accurate on a variety of languages
- Now we want to apply them to the VMS

Language Identification

- What language is the VMS?
- Apply the pattern decomposition method
  - Most accurate method resistant to anagramming
  - Compare to all 380 languages from our UDHR set

Results

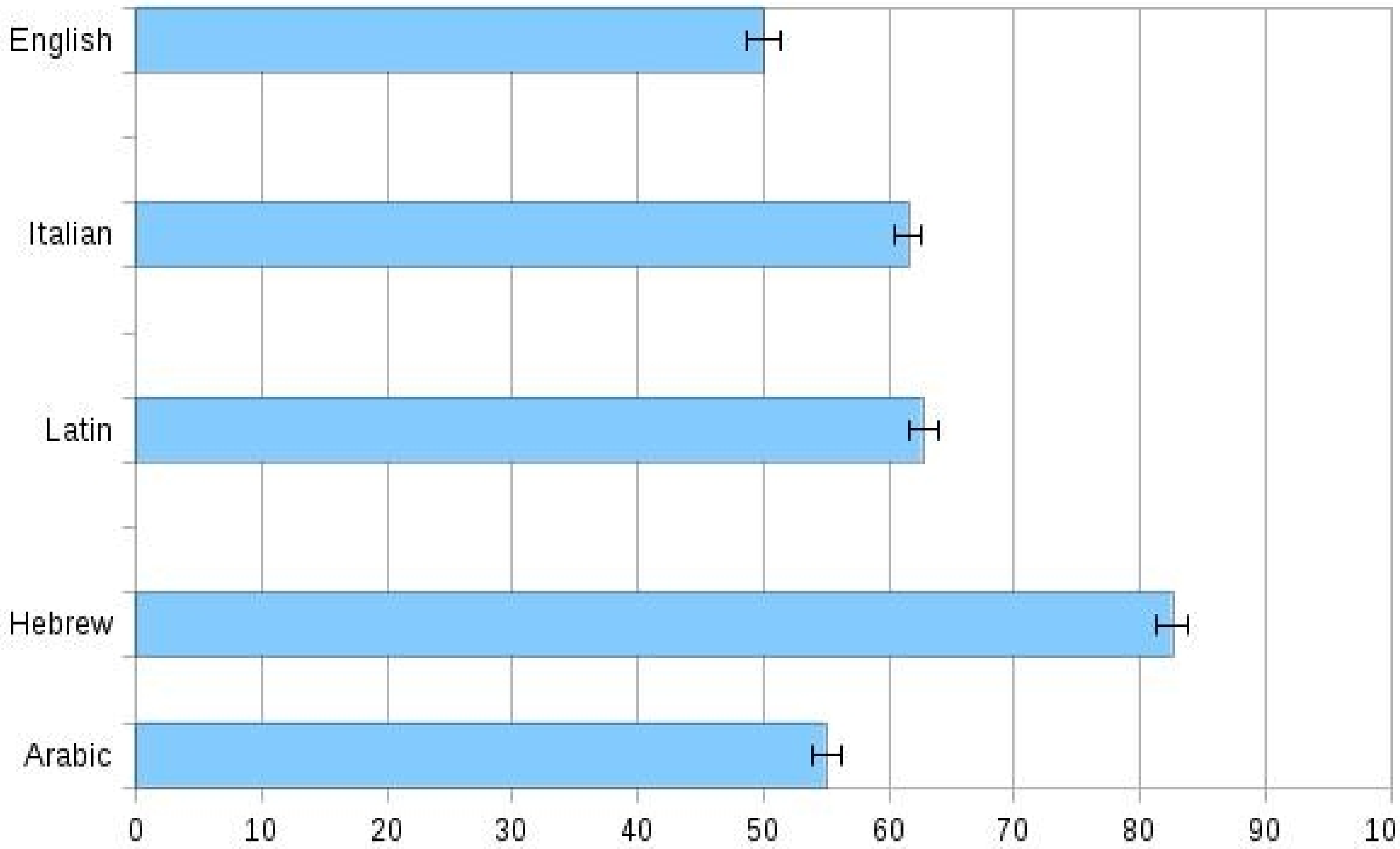


25
20
15
10
5
0
Languages
Hebrew
0.02
0.05
0.08
0.11
0.14
0.17
0.2
0.23
0.26
0.29
0.32
0.35
0.38
0.41
0.44
0.47
0.5
Distance

What if we try to decipher the VMS?

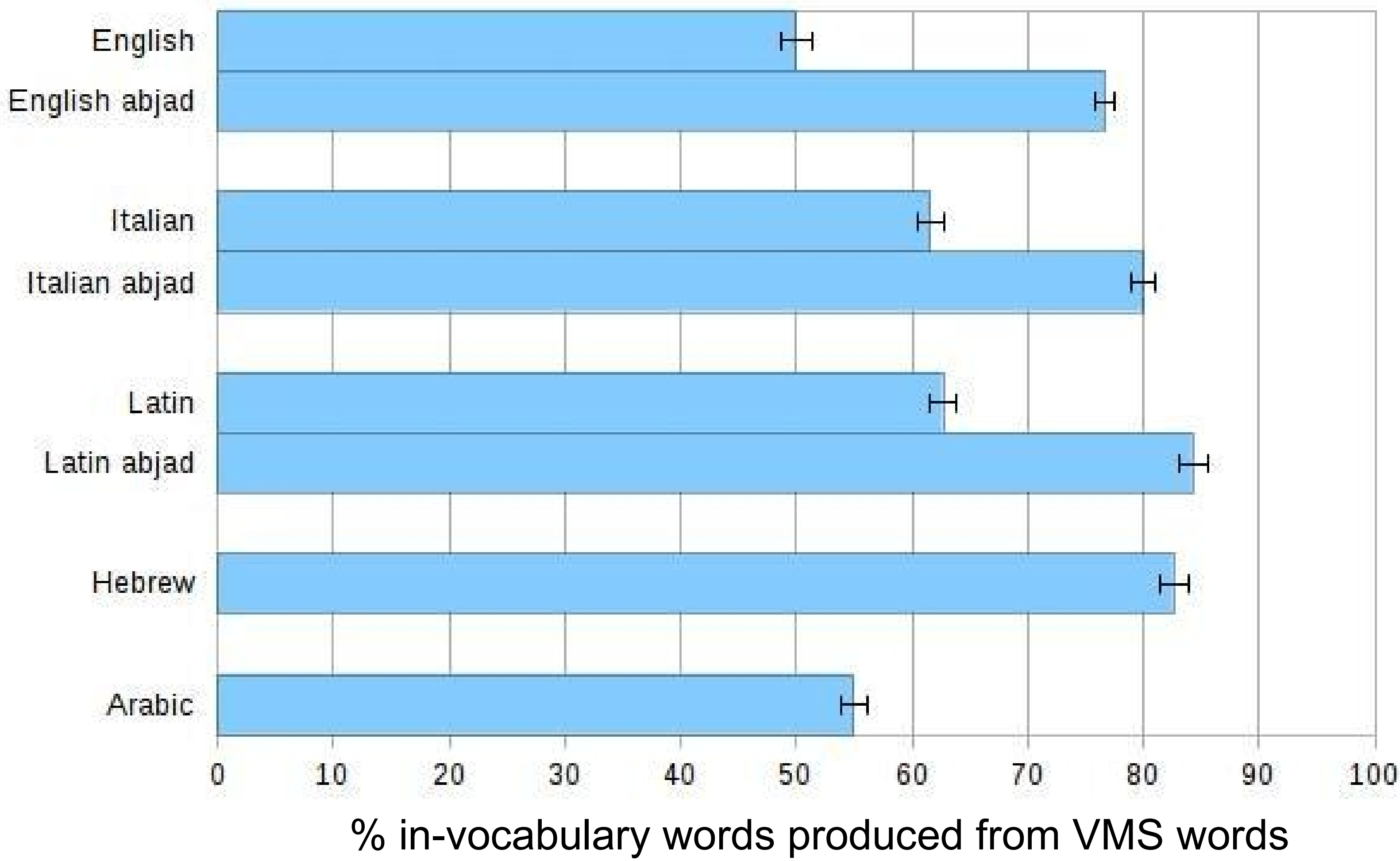
- Apply our pipeline to the VMS
- For each of 10 VMS pages, have our pipeline try to decipher it into English, Italian, Latin, Hebrew, and Arabic
- Count the number of in-vocabulary words in the output
  - How many words in each language were generated from the VMS using substitution and anagramming

Results



% in-vocabulary words produced from VMS words

Results


English
English abjad
Italian
Italian abjad
Latin
Latin abjad
Hebrew
Arabic
0
10
20
30
40
50
60
70
80
90
100
% in-vocabulary words produced from VMS words

Conclusion

- New substitution cipher solver
  - Outperforms previously published solvers
- Can extend solver to handle anagramming
  - 94% word accuracy
- Can accurately identify ciphertext language
  - 97% accuracy with best method
- Voynich manuscript
  - Evidence for Hebrew as source language

Thank you.

Hebrew Output Examples

- Input: VAS92 9FAE AR APAM ZOE ZOR9 QOR92 9 FOR ZOE89 (the first line of the VMS)
- Output: אנשיו עלי ו לביחו אליו איש הכה לה ועשה המצות
- Not quite a coherent sentence.
- Google Translate: *She made recommendations to the priest, man of the house and me and people.*
- Input from the Herbal section of the VMS (72 words)
- Output includes: הצר ‘narrow’, איכר ‘farmer’,

  אור ‘light’, אויר ‘air’, אש ‘fire’

Future Work

- Future work:
  - Unsupervised transliteration, transliteration mining
    - Non-alphabetic scripts
    - Unrelated languages
  - More reliable deniable encryption
  - Unified generative model
  - Analysis of "lost languages"