Character Encoding Detection with a Known Language

Jan Pomikálek, Vít Suchomel

RASLAN 2011
December 2-4, 2011

CHARACTER ENCODING

- text stored in a computer = a sequence of bytes
  - example: **9e** ed **9e 61** 6c **61** 20 73 74 6f 6a ed 20 35 **80**
- character encoding = mapping between bytes and characters

character	windows-1250	iso-8859-2	utf-8
a	61	61	61
ž	9e	be	c5 be
€	80	n/a	e2 82 ac

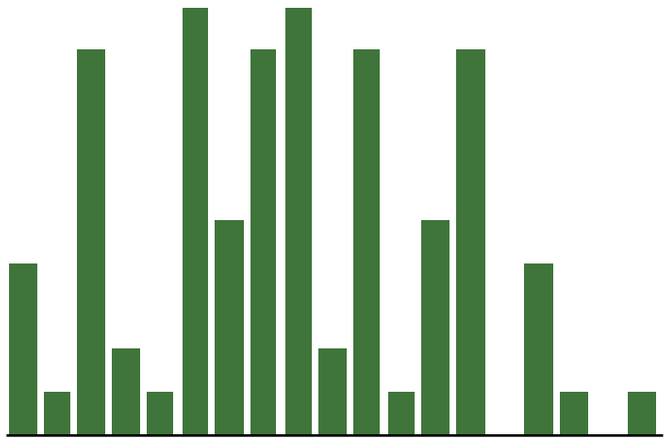
- *žížala stojí 5€*

CHARACTER ENCODING

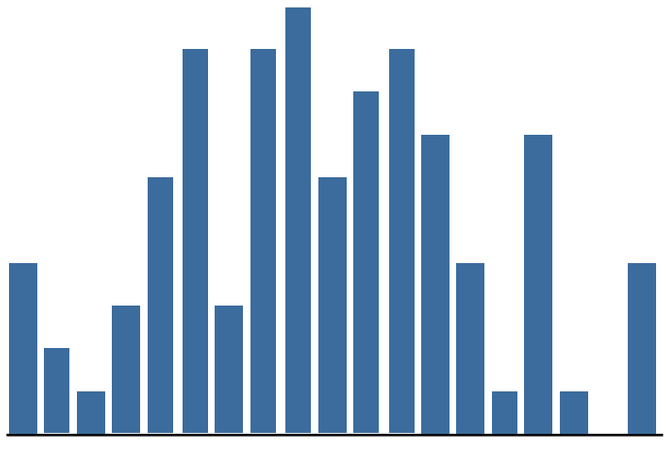
- bytes
  - 70 c5 99 c3 ad 6c 69 c5 a1 20 c5 be 6c 75 c5 a5 6f 75 c4 8d 6b c3 bd 20 6b c5 af c5 88 20 70 c4 9b 6c 20 c4 8f c3 a1 62 65 6c 73 6b c3 a9 20 c3 b3 64 79
- in windows-1250
  - pĽ™A-liĽ˘ ĽĺluĽĄouAŤkA˝ kĽŻĽ pA›l ÁŽAˇbelskA© Áły
- in iso-8859-2
  - pĽĂ-liĽĄ ĽžluĽĽouÄkĂ˝ kĽŻĽ pÄl ÄĂĄbelskĂŠ Ăły
- in utf-8
  - příliš žluťoučký kůň pěl ďábelské ódy

Automatic encoding detection

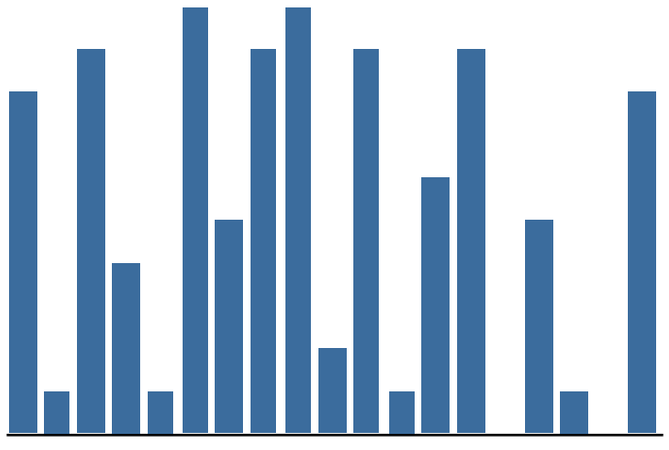
- byte frequency vector for the input text



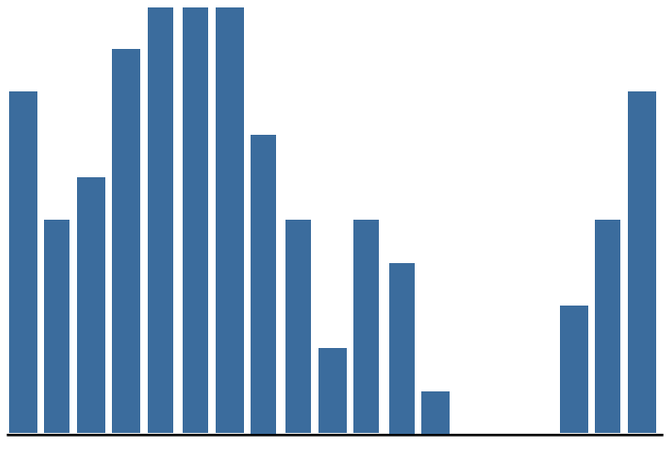
- compare with model vectors (scalar product)



iso-8859-1



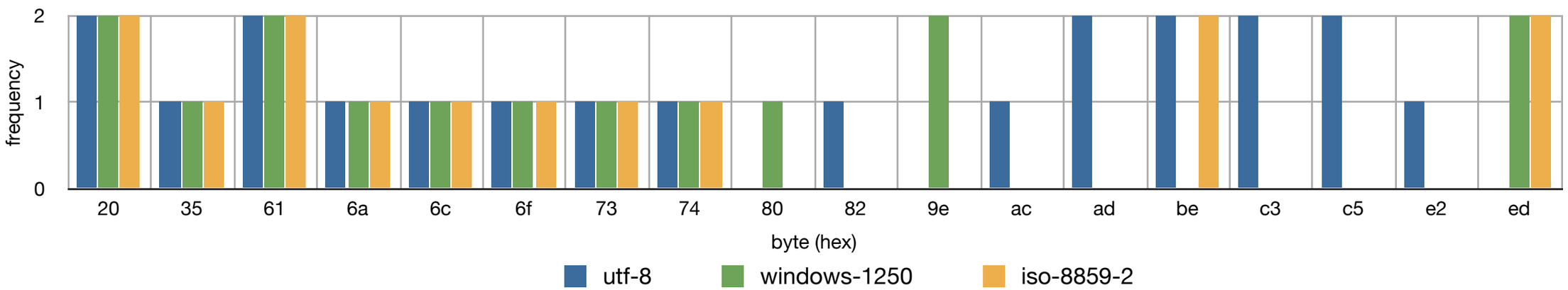
koi8-r



utf-8

CREATING MODELS (FOR A LANGUAGE)

- take a text in a known encoding
  - *žížala stojí 5€*
- convert to all encodings (commonly) used for the language
  - **utf-8:** c5 be c3 ad c5 be 61 6c 61 20 73 74 6f 6a c3 ad 20 35 e2 82 ac
  - **windows-1250:** 9e ed 9e 61 6c 61 20 73 74 6f 6a ed 20 35 80
  - **iso-8859-2:** be ed be 61 6c 61 20 73 74 6f 6a ed 20 35
- create byte frequency vectors



- prune items with the same frequency in all models (for efficiency)

TRAINING DATA

- take ca. 1000 web pages with texts in the target language
  - (collected with Corpus Factory)
- extract encoding information from meta tags
  - <meta http-equiv="Content-Type" content="text/html; charset=**UTF-8**" />
  - discard pages for which encoding cannot be determined
- find out which encodings are frequently used (on the Web)
  - e.g. for Czech:: **utf-8**: 60.2%, **windows-1250**: 32.2%, **iso-8859-2**: 6.0%
  - ignore encodings with freq < 0.5%
- convert all pages to all frequently used encodings
- create frequency vectors

WHY LANGUAGE SPECIFIC MODELS?

- reduces number of classes (encodings)
- a byte may correspond to several different characters in different encodings which have different usage frequency in different languages
  - example: byte a3
    - £ (pound symbol) in windows-1252
    - Ł (L with stroke) in iso-8859-2
- using language specific models improves accuracy

TRIPLE-BYTE MODELS

- byte a9
  - © (copyright symbol) in windows-1250
  - Š (S with caron) in iso-8859-2
- both commonly used characters in Czech
- imagine a text with only ascii characters plus one a9 byte
  - impossible to classify correctly based on frequencies of single bytes
- using context helps, for instance:
  - “Šk” is more likely than “©k”
  - “© ” is more likely than “Š ”
- conclusion: use frequency vectors on tuples of bytes
- 3-grams produced better results than 2-grams in our experiments

EVALUATION

- 5-fold cross-validation on training data

	Czech		English		German		Greek		Italian		Norwegian	
	freq	accuracy	freq	accuracy	freq	accuracy	freq	accuracy	freq	accuracy	freq	accuracy
utf-8	60.2%	100.0%	56.9%	95.8%	54.6%	100.0%	68.5%	100.0%	54.2%	100.0%	63.0%	100.0%
windows-1250	32.2%	100.0%	0.3%	n/a	0.1%	n/a	0.2%	n/a	0.0%	n/a	0.1%	n/a
windows-1252	0.4%	n/a	9.4%	97.5%	6.5%	97.3%	3.1%	75.8%	7.1%	95.7%	7.0%	97.4%
windows-1253	0.0%	n/a	0.0%	n/a	0.0%	n/a	14.3%	99.3%	0.0%	n/a	0.0%	n/a
iso-8859-1	1.0%	89.5%	32.8%	90.9%	37.1%	85.8%	1.7%	71.2%	37.9%	85.1%	29.3%	88.2%
iso-8859-2	6.0%	99.6%	0.0%	n/a	0.1%	n/a	0.0%	n/a	0.1%	n/a	0.1%	n/a
iso-8859-7	0.0%	n/a	0.0%	n/a	0.0%	n/a	12.0%	97.2%	0.0%	n/a	0.0%	n/a
iso-8859-15	0.0%	n/a	0.0%	n/a	1.2%	85.6%	0.0%	n/a	0.0%	n/a	0.4%	n/a
training docs	801		668		773		879		771		740	
w. avg accuracy	99.2%		93.5%		93.7%		97.9%		93.3%		95.7%	

IMPLEMENTATION

- in Python
- open source (BSD License)
  - available from: http://code.google.com/p/chared/
  - online demo: http://nlp.fi.muni.cz/projects/chared/
- currently supports 51 languages
  - http://code.google.com/p/chared/source/browse/#svn%2Ftrunk%2Fchared%2Fmodels