Automatic Language Identification

Benjamin Schnieders, Florian van Daalen

June 24, 2013

Abstract

Automatic language identification is an important first step for natural language processing. This paper presents an approach that breaks down any text to sequences of 2 characters, so called bigrams, and analyzes their occurrences in different languages. Detection proceeds likewise, choosing the most likely language. It will show that, using statistical methods, even single bigrams can hint towards a language classification, and that by combining information of multiple bigrams, an extremely performant and accurate classifier can be built.

1 Introduction

Before language-specific natural language processing tasks such as stemmming or coreference resolution can be performed, the language of the text has to be determined. If there is no human tagging available, automatic translation engines need to detect the language themselves before translating. E-readers might assist readers, e.g. those who are fluently reading one language, but lack the needed understanding of another, in reading by detecting foreign language phrases in native language documents. These phrases can then be highlighted and, if the reader wishes, automatically translated.

1.1 Encoding detection

The Internet is a vast corpus of mostly natural language text. Through different machine types, operating systems and input editors, a number of encodings are in use, in certain cases even within one document. To perform language recognition successfully, the character encoding has to be known, and thus detected, as well. In order not to confuse the same languages with different encodings, or different languages that are only similar because they use the same encoding, the described language detection method performs language detection only in UTF-8. Other encodings will be detected as such, but their language will remain unknown until the text is transcribed to UTF-8 and re-classified. UTF-8 is the de-facto web-standard and the most compact form to store Unicode characters. As it is also downwards compatible to ASCII, all English, ASCII-encoded text can be fed into the language recognition system without modification.

As the recognition is based on character bigrams, UTF-8 characters, that can spread over 4 bytes, are broken up to multiple bigrams.

2 Text Input

To train the language recognition system, text corpora with known languages have to be analyzed, and the probability of each character bigram to occur in each language is calculated and inserted into a table. Ideally, training text should only contain words and sentences of one language each. Different language words, used symbols (such as parentheses, hyphenations and numbers) form noise in the training set. Also, the training text should contain as much running text as possible, with no named entities mentioned.

2.1 Wikipedia as a Text Source

Wikipedia is an online encyclopedia maintained by its users. Users can make an account and then adapt or create pages in the encyclopedia about any subject. Since users from all over the world contribute it grows rapidly, and is updated in a variation of languages, ranging from English & Japanese to Esperanto and local dialects such as Limburgs. Due to the open nature and ability of average people to adapt pages it is not always reliable as people occasionally will include wrong information out of spite, as a joke, or just being mistaken. However, due to the peer reviewed system Wikipedia employs these mistakes are usually quickly found and fixed, after which the perpetrators might be blocked depending on the severity of their misbehavior. It should be noted that the reliability of the different pages differs per subject and per language. An unpopular subject or a page in an unpopular language will be reviewed less often and thus mistakes might linger longer.
Wikipedia was chosen as a source for input text because of a number of reasons. First of, as an encyclopedia, it has a massive amount of text annotated with which language it is in. Thus making it easy to generate a large corpus from it for the purpose of training a language recognition model on it.
Secondly due to the international nature it contains information about the same subject in a large variety of languages. Making it easier to keep the data set somewhat homogeneous across languages with respect to subject matter. Since it is simple to guarantee the same subject in multiple language, it is easy to represent the fact that some aspects will be shared among languages, thus creating a more realistic database.
Thirdly since the languages share their subject matter, to a degree at least, the model will not accidentally distinguish languages based on jargon or names. For example if the text for two languages were about vastly different subjects, say physics and philosophy, it might be the case that the model will distinguish between the two text based on the words ”Kant” and ”Newton”, as neither of those will appear in both texts. However had both samples been about the same subject this would not have happened.

Furthermore, as an online encyclopedia, Wikipedia articles regularly includes abnormalities one would not find in normal bodies of text. A number of examples would be links towards other pages, footnotes, mathematical formulas & pseudo-code. This creates some interesting noise. Also, as an encyclopedia, the articles are standardized to a degree and do follow some rules, which does guarantee a certain degree of similarity between each language with respect to formatting and formality. For example slang will not be used nor will you get a weird lay-out as one might get in a flyer. Besides its web nature, Wikipedia pages contain a significant level of noise, being numbers, image captions and a non-ideal writing style: Being encyclopedic articles, a lot of enumerations, sentence fragments and short sentences are used, but floating text would be preferable. On more ideal training corpora, language identification might produce better results. However, this might lead to a poorer generalization performance as the real world is not without noise.

2.2 Character Bigrams as Text Representation

Training text and classification examples are broken up to sequences of two characters on byte level. This technique is inspired by the work of Cavnar and Trenkle, [1].

”example” => ” e”, ”ex”, ”xa”, ”am”, ”mp”, ”pl”, ”le”, ”e ”

Note that at the text at the beginning and end are padded with a null byte, visualized here by a space. This is not strictly necessary for training, but for recognition it is: The first character might already contain valuable information, and thus should not be discriminated by using its information just once, as opposed to twice for any other character.

UTF-8 characters might be broken up in this process, as the German word ”Fräse” (engl. milling machine) will demonstrate:

”Fräse” => ” F”, ”Fr”, ”r_”, ”___”, ”_s”, ”se”, ”e ”

Unprintabe characters are replaced with an underscore here. Although this poses a limitation, practically this should only influence correct classification in the case of two similar languages that share most representations in the follower bytes of UTF-8 characters. This is highly unlikely.
As there are only 65535 distinct character bigrams, one can easily build a table holding the probabilities for each one to occur in any language. For two languages, using double precision (8 bytes per probability) only a megabyte of memory is used; using 20 languages as the system described in this paper does, 10 megabytes of memory are needed. Working on all 4 billion possible UTF-8 characters is inapplicable for most systems.

2.3 Information Gain per Bigram

To model natural language as accurately as possible, a sufficient training text length is needed. In theory, the information gain with every added bigram

should rise monotonically, however, the growth is logarithmic. This finding is in correspondence with Heap's law, postulating logarithmic dictionary growth for growing datasets. For practical purposes, a cutoff has to be found. Analyzing the existing training text corpora, Figure 1 shows that after about 300.000 characters, no more new bigrams are found. Of course, the weights for the already introduced bigrams are still affected, but the assumption that already a small portion of text can successfully model text of a language is supported. Hence, only a comparatively small training corpus was used per language.

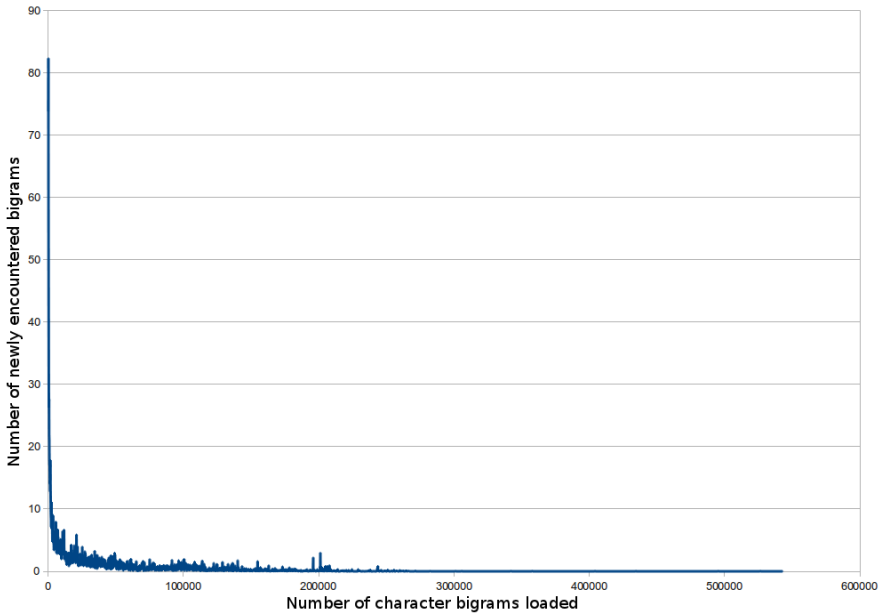

Figure 1: The number of previously unknown bigrams with growing training text length.

2.4 Languages

Some of the most-spoken languages were chosen for demonstration. This is the complete list of languages used, along with the native writing of that language:

English: English

German: Deutsch

French: Français

Esperanto: Esperanto

Danish: Dansk

Croatian: Hrvatski

Greek: Ελληνικά

Italian: Italiano

Japanese: 日本語

Korean: 한국어

Dutch: Nederlands

Russian: Русский

Spanish: Español

Arabic: العربية

Chinese: 中文

Hindi: हिन्दी

Portuguese: Português

Vietnamese: Tiếng Việt

Additionally, the language identification system uses two more languages internally, one to describe non-UTF-8 text, and one to classify text that has a low resemblance with any language - if the highest probability is below a certain threshold, a bigram or a whole text part is classified as *unknown*.
For each language, about 500 kilobyte of training text were extracted from Wikipedia. Note that Chinese here depicts Standard Chinese with simplified characters.

From the resulting probability table, one can already calculate similarities between languages. As of all 65535 possible bigrams, only 18891 could be parsed from the training set (about 28.82 %), the resulting language descriptor vectors are relatively sparse. A good measure to compare sparse vectors is by using the cosine distance. A plot of cosine similarities between vectors of all languages can be found in Figure 2
From this chart, one can easily read off that languages using a different alphabet can be easily separated, as they share no resemblance. Japanese text, with Japanese characters being developed from Chinese characters, has some resemblance with Chinese text. The hardest task is obviously differentiating between all languages using the same alphabet. Danish and Dutch as well as Portuguese and Spanish have a very high similarity, it is expected that classification is difficult here.

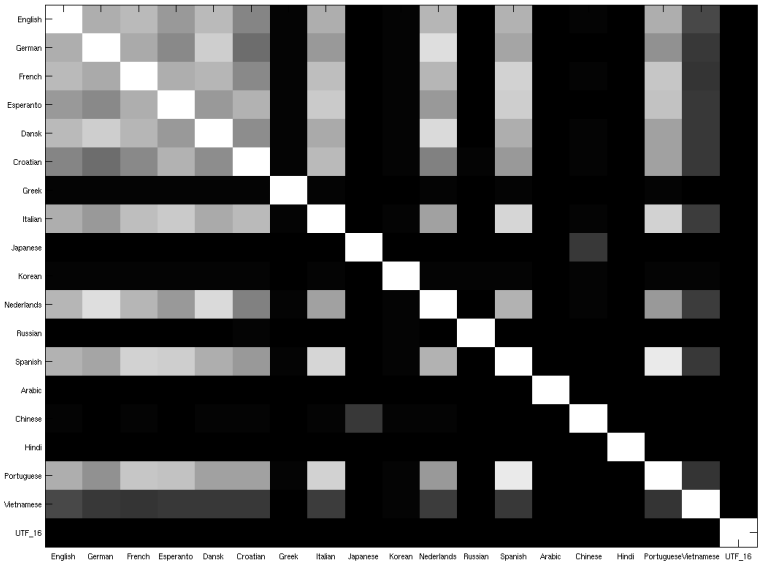

Figure 2: A resemblance map using the cosine similarity between each pair of languages. Color ranging from black, zero resemblance, to white, full resemblance.

3 Language Detection

Training the classifier is done by counting how often any bigram is recorded for any language. As the table can be indexed in constant time, both training and classification are extremely fast.
From example text, a vector representation in bigram space can be obtained using the same procedure as during classification. This vector is then compared to the existing language vectors, and the closest match is chosen. However, this method only works if the full text is known to be of just one language; for multi-language texts the following approach is used.
After training, the amount of bigrams recorded for each language is taken and all data scaled by the maximum; this ensures that all languages can receive equal weight. Otherwise, languages that receive more training would achieve higher probabilities. Next, each bigram is normalized, making its entries proper descriptions of the probability that this bigram was actually drawn from a training text of a certain language.
Now, a likelihood vector can be retrieved from the probability table for each bigram. Using this approach on example text yields unsatisfying results, as most single bigrams do not give a strong hint for the correct language in the more advanced, same alphabet, language identifications. See Figure 3 for a visual example. On distinct languages using a different alphabet however, this method

works to a satisfactory degree. To achieve more consistent classifications, an averaging window can be moved over the text to be classified, drastically reducing classification noise. Figure 4 shows the result on the same example text with an averaging window of 100 bigrams. Combining the likelihood vectors of multiple bigrams uses more information to classify, and thus is more accurate.

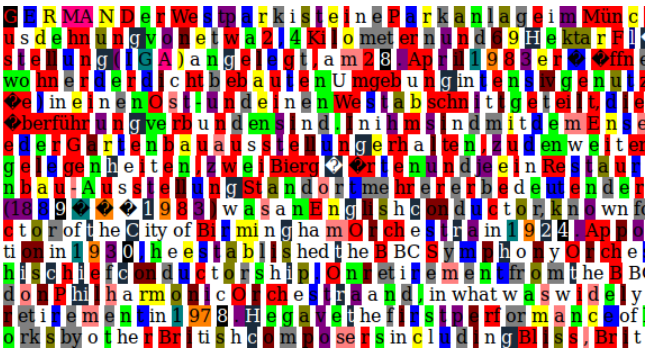

Figure 3: Classification noise, as single bigrams occur in many languages equally often.

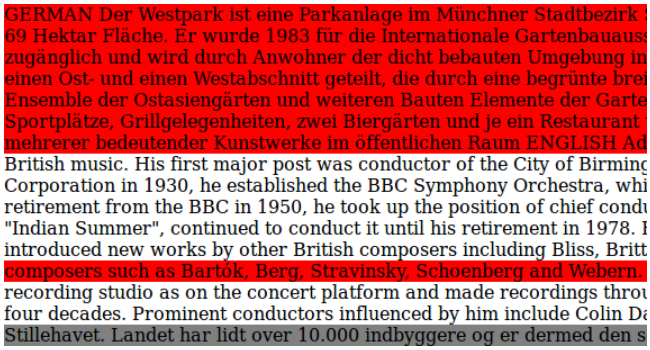

Figure 4: Overcoming classification noise by assigning the weighted average classification of 100 bigrams with a moving average window.

3.1 Encoding Detection

Either learning sets have to span all possible encodings, which is hard to achieve, or all training data and examples should use the same encoding. Being most common in web environments, the chosen charset was UTF-8. The classifier was trained on 19 UTF-8 encoded languages and a large amount of files with different encoding formats, combined under the label UTF-16. The classifier can then determine whether the encoding of any text is valid, rejecting text that is not UTF-8. Such text can then be converted and re-classified.

4 Results

Example text was extracted from Wikipedia to produce a test set. Besides its shorter nature, the control text has the same features than the training data, with respect to noise, entity names and numbers. Language identification was performed on both a single bigram level, as well as on longer substrings of 100 bigrams. The longer text samples represent the more real world applications, in which written sentences shall be classified. The single bigram results reveal interesting insight on the theoretical minimum length of text to be classified to be classified correctly with a given accuracy.

4.1 Single Bigram Classification

All occurring bigrams in the test set were classified and the results entered into the confusion matrix that can be found in Appendix B.1. From the confusion matrix, measures like precision and recall can be calculated. Excellent results were obtained for languages with a distinct alphabet, for example Greek with a precision of 0.9168 and recall of 0.9257. For languages sharing the same alphabet, results were worse. A complete list of precision and recalls for all languages can be found in Appendix B.2.
The correlation between precision and recall is coherent, as shown in Figure 5. There is one interesting outlier, caused by the classifier being able to recall unsupported encodings quite well (0.9899), but featuring only a lower precision (0.6638). This might be caused by many single bytes that form valid encoded characters but are trained as unsupported encoding.

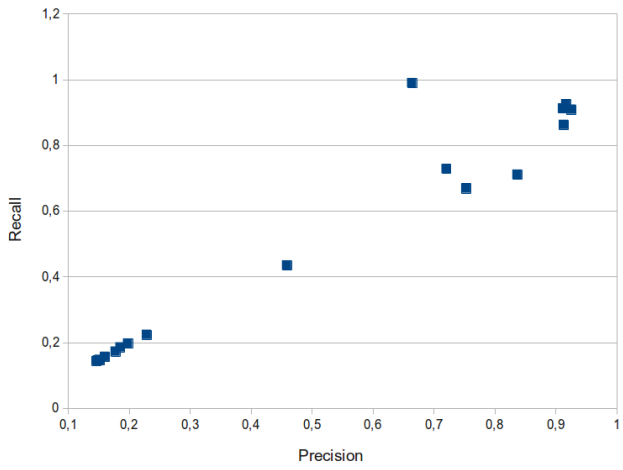

Figure 5: Precision plotted vs. recall, note the outlier caused by different encodings.

The accuracy of the whole classifier on single bigrams is 0.4646. As there

are 19 positive classes, the classifier still performs about 8.8 times better than random. Any language can be classified by a single bigram better than random. As already explained in Section 3, usually example texts to classify are a lot longer than merely two characters. Using the combined information of multiple bigrams raises the accuracy, as the probability for a misprediction is repeatedly multiplied with itself. For a classifier performing better than random and n bigrams, the precision on n bigrams can be estimated with

$$precision_n = 1 - \prod_{i=1}^{n}(1 - p_i) = 1 - (1 - p)^n$$

where p denotes the precision of the classifiers to classify the correct language for bigram i.

Using this formula, an estimation of the minimal length of example text to be classified with any given accuracy can be made.

$$minimal\_length = log(1 - precision\_to\_achieve)/log(1 - precision)$$

This calculation estimates that about 28 bigrams are needed to identify English with a precision of at least 0.99. However, this only holds if each bigram is equally informative, which in this case means it should be drawn randomly. Natural text though tends not to have a uniform character und thus bigram distribution. To compensate for possibly occurring deviations, the Shannon Entropy is calculated for each bigram, yielding an estimate of how probabilistically random bigrams occur in each language. A table of entropies for each language can be found in Appendix A.2.

Multiplying the number of bigrams needed with the actual information any bigram hold, that is, the entropy value of the corresponding language, a better estimate of the minimum number of bigrams needed to achieve any accuracy can be calculated. According to the average over all languages, at least 20 bigrams are needed to achieve 0.99 precision. For Italian, Spanish and Portuguese though, at least about 40 bigrams are needed. Still, only statistical frequency of bigrams occurring is considered, not the fact that natural languages follow a certain structure, that is, the occurrence probability of any bigram is not independent from the preceding bigrams.
Of course, the accuracy of the classifier can be improved by eliminating impossible languages. If in an English text, only French passages should be highlighted, the classifier can use that information and distinguish only those two.

4.2 Sequence Classification

The assumption that longer sequences are classified with a higher accuracy was supported empirically. Figure 6 shows that beginning from the value obtained for single bigrams, the accuracy rises, converging to 1 with increasing example text length. Although the earlier calculation predicted a 0.99 accuracy for text

lengths of in average 20 bigrams, this level of accuracy is only reached for texts lengths of 100 bigrams. This is the effect of bigram sequences not being randomly sampled; instead, natural language sequences show to follow a certain structure, which prevents giving each bigram full expressiveness.

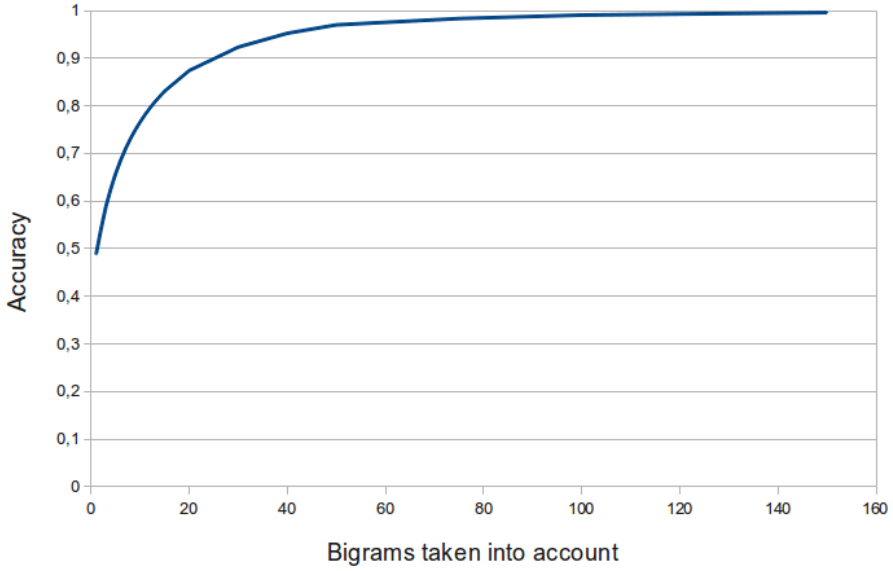

Figure 6: Accuracy for growing example text lengths.

A full confusion matrix for test text sequences of 100 bigrams can be found in Appendix C.1. Note that still languages like Spanish and Portuguese are confused in rare cases. Precision and recall are close to 1 for all languages, as the accuracy is 0.9906 for this setup.

4.3 Dictionary Compression

For mobile or embedded devices like e-readers or mobile phones, using up 10 megabytes of memory to classify out of 20 languages might be unacceptable. Therefore, the classifier needs to be compressed in a way to retain most of its accuracy while possibly taking longer to classify.
TF-IDF was calculated for each bigram per language. A high TF-IDF value means, that the selected bigram is very expressive about this language. For example, the 6 top-ranked bigrams for English are:

- 12.7794, ry
- 12.2195, 's
- 12.1429, Ea

- 12.0469, ly
- 11.7374, th
- 11.1011, ty

These are two-character combinations that occur often in English text, but not often anywhere else. For Dutch, the top ranking bigrams are:

- 13.5132, ee
- 12.6652, oo
- 11.1048, aa
- 9.84553, oe
- 9.44828, gd
- 8.26087, Aa

More characteristic bigrams for selected languages can be found in Appendix A.1. Selecting only the top-scoring bigrams for classification, one can compress the classifier without losing much accuracy. Figure 7 shows that accuracy only

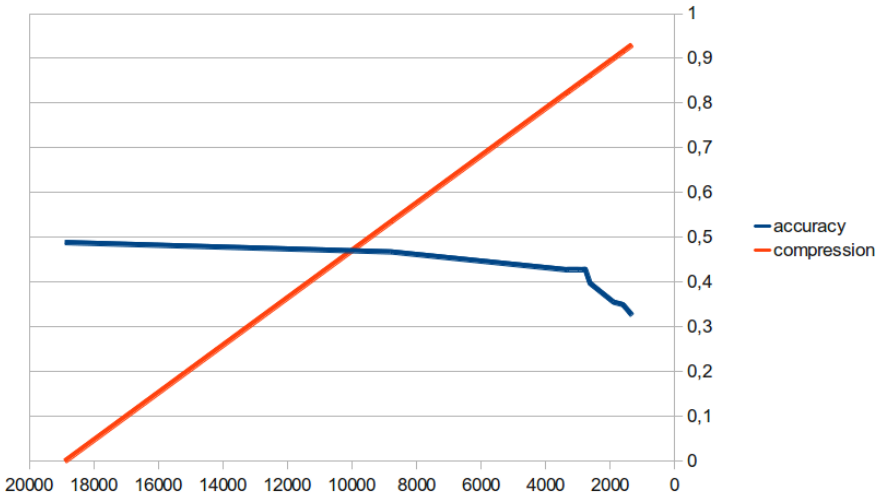

Figure 7: Accuracy and compression level plotted for various dictionary sizes. Bigrams were selected using TF-IDF.

drops slowly until a critical point is reached, at a compression level of about 0.85. Beyond that point, accuracy drops quickly. If longer texts should be classified, i.e., if the averaging window is allowed to be wider, a reduction to 15% of

the memory might be a welcome alternative for low-memory devices. Logarithmic search on about 2800 instances should be negligible calculation overhead compared to the memory savings.

5 Conclusions

The results clearly show that even on the low level of character bigrams, languages can be quite well statistically differentiated from another. However, to achieve an arbitrary performance, text samples to be classified have to be of a certain size, below which suboptimal results are achieved.

5.1 Future Work

Currently, averaging over multiple bigrams to improve classification precision leads to slightly misplaced classification boundaries. This means for example that if English text, which is comparably hard to identify, is followed immediately by Greek language, the seam between both languages will be in the English text, as the averaging window already picks up some strong hints towards Greek. This might be countered by using a nonlinear average, giving bigrams farther away from the bigram to classify less weight. Another possibility would be to perform an optimized seam placement with re-classification of near-border text. Whenever, using an average window, a language change is detected, the area around the chance is re-classified with the classifier set to only allow the two languages contributing to the chance. Then, the midpoint between the leftmost new language and the rightmost old language is chosen for the seam.
Further research can be done in the field of improving the classifier for very short sequences, for example by using larger training corpora including less noise. As compression of the bigram table using TF-IDF was very successful, one can expect success also on character trigrams or quads. Combining bigram probability tables with selected character trigrams and quads might improve the classification accuracy even on extremely short character sequences.

References

[1] William B and John M. Trenkle Cavnar. N-Gram-Based Text Categorization. *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, 1994.

A Language Properties

A.1 Significant Bigrams

TF-IDF ranked bigrams for selected languages:
German:

- 12.8716, ah
- 11.7817, ü
- 11.4621, ei
- 10.7727, Ei
- 10.5857, tz
- 10.5547, ö

French:

- 11.4454, iq
- 11.2785, x
- 11.1331, é
- 10.2065, ux
- 9.07752, d’
- 8.85268, è

Greek:

- 15.9218, α
- 15.9214, ι
- 15.8903, ο
- 15.8537, λ
- 15.8468, σ
- 15.8119, ί

A.2 Language Entropy

Language	No. different bigrams	Entropy
English	2567	0.730573
German	2462	0.738945
French	2806	0.722866
Esperanto	2454	0.721849
Dansk	2359	0.743117
Croatian	2346	0.751398
Greek	2275	0.627021
Italian	2410	0.712775
Japanese	4922	0.740082
Korean	3974	0.735869
Dutch	2501	0.723475
Russian	2323	0.611161
Spanish	3260	0.701184
Arabic	1288	0.635658
Chinese	5720	0.803987
Hindi	1822	0.552861
Portuguese	2506	0.722114
Vietnamese	3099	0.704143
UTF_16	7572	0.531062

Number of different bigrams recorded for all languages and their entropies.

B Single-Bigram Results

The following are results obtained on classifying single bigrams, i.e., classifying the shortest possible sequence.

B.1 Confusion Matrix

Confusion:	Engl.	Ger.	Fr.	Esp.	Dansk	Croat.	Greek	Ita.	Jap.	Kor.	Dutch	Rus.	Span.	Arab.	Chin.	Hindi	Port.	Viet.	UTF_16
English	497	347	281	250	320	206	16	276	11	21	329	8	279	7	9	14	272	199	15
German	293	665	270	217	345	206	13	244	8	22	392	8	240	10	5	15	235	153	15
French	284	240	625	245	255	213	30	297	3	18	248	12	331	15	4	9	340	181	6
Esperanto	258	230	269	582	272	358	10	328	3	6	268	3	301	4	2	7	300	126	30
Dansk	308	327	285	264	526	284	11	266	6	14	358	9	248	6	3	12	242	155	32
Croatian	238	249	209	343	277	753	16	256	16	27	257	11	234	11	10	15	242	146	45
Greek	3	3	6	3	4	5	3107	6	4	31	4	32	4	64	4	29	5	17	26
Italian	286	246	320	328	237	294	12	496	3	8	239	3	364	11	1	15	356	133	6
Japanese	5	4	1	4	4	3	2	3	2446	49	2	5	1	6	614	3	1	12	190
Korean	10	16	17	8	12	10	16	13	28	2388	14	49	9	56	31	41	9	36	592
Dutch	317	357	258	278	344	283	13	248	7	23	529	8	245	10	7	11	241	163	14
Russian	5	5	6	5	6	8	42	8	4	68	3	3051	4	24	8	38	4	11	56
Spanish	272	249	325	321	245	247	11	353	3	6	246	5	491	7	1	8	397	161	10
Arabic	2	3	14	4	3	2	34	4	5	41	3	23	4	3063	5	26	8	25	87
Chinese	2	1	3	1	1	1	3	1	825	35	1	9	2	2	2246	4	2	13	205
Hindi	2	1	2	1	6	3	19	4	7	46	1	41	2	34	12	2895	2	17	259
Portuguese	292	261	312	291	261	257	13	330	5	12	257	5	372	7	3	11	487	177	5
Vietnamese	211	159	170	140	166	164	20	158	8	35	154	12	163	23	15	17	188	1461	89
UTF_16	3	1	3	1	3	2	1	1	2	4	3	1	2	1	4	1	1	1	3322

Confusion Matrix for classifying single bigrams. Floating point values were rounded for better readability.

B.2 Precision & Recall

Language	Precision	Recall
English	0.1511	0.1481
German	0.1975	0.1980
French	0.1850	0.1861
Esperanto	0.1771	0.1734
Dansk	0.1600	0.1566
Croatian	0.2284	0.2244
Greek	0.9168	0.9257
Italian	0.1506	0.1477
Japanese	0.7204	0.7288
Korean	0.8367	0.7115
Dutch	0.1599	0.1576
Russian	0.9257	0.9091
Spanish	0.1489	0.1462
Arabic	0.9114	0.9127
Chinese	0.7527	0.6691
Hindi	0.9131	0.8628
Portuguese	0.1463	0.1452
Vietnamese	0.4586	0.4355
UTF_16	0.6638	0.9899

Precision and recall for all languages.

C Longer Sequence Results

These results were obtained on classifying character strings consisting of 100 bigrams.

C.1 Confusion Matrix

Confusion:	Engl.	Ger.	Fr.	Esp.	Dansk	Croat.	Greek	Ita.	Jap.	Kor.	Dutch	Rus.	Span.	Arab.	Chin.	Hindi	Port.	Viet.	UTF_16
English	3731	444	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
German	0	4175	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
French	0	0	4175	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Esperanto	0	0	0	4175	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dansk	0	0	0	0	4062	112	0	0	0	0	0	0	0	0	0	0	0	0	0
Croatian	0	0	0	0	0	4175	0	0	0	0	0	0	0	0	0	0	0	0	0
Greek	0	0	0	0	0	0	4175	0	0	0	0	0	0	0	0	0	0	0	0
Italian	0	0	0	0	0	0	0	4175	0	0	0	0	0	0	0	0	0	0	0
Japanese	0	0	0	0	0	0	0	0	4175	0	0	0	0	0	0	0	0	0	0
Korean	0	0	0	0	0	0	0	0	0	4175	0	0	0	0	0	0	0	0	0
Dutch	0	0	0	0	0	0	0	0	0	0	4175	0	0	0	0	0	0	0	0
Russian	0	0	0	0	0	0	0	0	0	0	0	4175	0	0	0	0	0	0	0
Spanish	0	0	0	0	0	0	0	0	0	0	0	0	4044	0	0	0	131	0	0
Arabic	0	0	0	0	0	0	0	0	0	0	0	0	0	4175	0	0	0	0	0
Chinese	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4175	0	0	0	0
Hindi	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4175	0	0	0
Portuguese	4	49	0	0	0	0	0	1	0	0	0	0	0	0	0	0	4121	0	0
Vietnamese	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4175	0
UTF_16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4175

Confusion Matrix for classifying strings containing 100 bigrams. Floating point values were rounded for better readability.