

TEXT RECOGNITION MEMORY

PAWEŁ MATYKIEWICZ, WŁODZISŁAW DUCH

ABSTRACT. This paper describes a data base and an algorithm that operates on this data. The purpose of this algorithm is to give a powerful tool for recognition of properly written sentences and correction of poorer sentences and misspelled words.

1. THE IDEA

The idea of the algorithm is based on the hypothesis of the function of cerebral cortex that was described in [3]. Author presented a computational method that enables to anticipate last word of a given phrase (e.g. cars drove down a [lane, freeway, boulevard]). He used 10 000 common English words and correlations between them after training of 1.4 billion word proper English training corpus. The correlations between words were collected by looking from a root word up to four position forward.

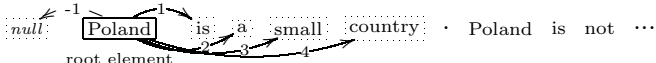
In this paper extension of this method is presented. Novel approach enables a correction of badly written sentences. It is possible to find not matching words (misspelled or at wrong position) inside a sentence and change it to a proper ones. This approach uses not only forward correlation but also backward. More over the same method of training that is applied to sentences (sections, paragraphs) is also applied to tri-grams¹ [1] of every new word that occurs during training process.

2. LEARNING

First a learning method is presented. For explanatory purpose we introduce a small training corpus:

- (1) Poland is a small country. Poland is not a city. Warsaw is not a country.

2.1. **Sections learning.** *Main unit* during the training process is a section (paragraph). In case of corpus 1 there is only one section. *An Element* is a word or any sentence punctuation. After obtaining a single section from a corpus first element is considered as a root element (as in example 2). All needed correlation should be written down. In the case of the correlation to utter elements only one correlation to so called "null" is noted.

- (2)  The diagram shows the sentence "Poland is a small country" with "Poland" enclosed in a box and labeled "root element" below it. Arrows indicate correlations: a dashed arrow labeled "-1" points from "Poland" to "null:" on the left; a solid arrow labeled "1" points from "Poland" to "is"; a solid arrow labeled "2" points from "Poland" to "a"; a solid arrow labeled "3" points from "Poland" to "small"; and a solid arrow labeled "4" points from "Poland" to "country".

The following table should be obtained after parsing the first element of the first section of introduced corpus 1 : Table 1 can be also understand as a sparse 3-dimensional matrix of a size: $words_{coprus} \times (forward - backward) \times words_{coprus}$,

Date: 05.07.2004.

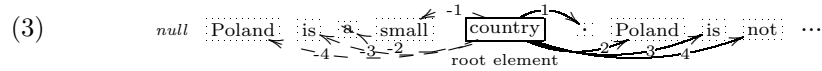
¹Alternatively bi-grams can be used.

root element	relative position	branch element	number of occurrence
Poland	-1	<i>null</i>	1
Poland	1	is	1
Poland	2	a	1
Poland	3	small	1
Poland	4	country	1

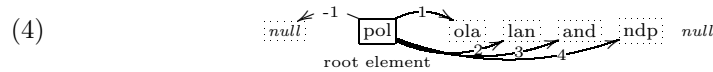
TABLE 1. Parsed the first element of the corpus 1 to a table

where $words_{coprus}$ is a number of different words from training corpus, $forward \in \mathbb{N}$ is a distance to the last correlation and $backward \in \mathbb{Z} \setminus \mathbb{N}$ is a distance to the first correlation.

After parsing first element next element is taken and so on. Sentence punctuation is also included because it can help with correction of badly punctated sentences.



2.2. Words learning. The same method is applied to the words, but this time *main unit* is a word and *an element* is a tri-gram. For an improvement of a word recognition also a tri-gram of two last letters and first letter is included. So first sub-unit for introduced corpus 1 will be "poland" (in the case of words capital letters are changed to lower case):



It is possible to get a similar table to 1 table. After parsing first element "pol" we

root element	relative position	branch element	number of occurrence
pol	-1	<i>null</i>	1
pol	1	ola	1
pol	2	lan	1
pol	3	and	1
pol	4	ndp	1

TABLE 2. Parsed the first element of the word "poland" to a table

move one step further and so on until all correlation are written down for every element. For n letter word there is $n - 1$ tri-grams. A 2-letter word e.g. "is" will have tri-gram "isi". In order not to loose 1-letter words a tri-gram is set to correspond to that letter.

2.3. Files. Tables 1 and 2 are stored separately in two different files. More over elements are replace with indexes. So indexes for elements in sections have they own file and elements in words also. Indexes file is organize in the following way:

| element | index | occurrence |

and the data base file:

| root element | relative position | branch element | number of occurrence |

A small piece of indexes and data base file for **sections** after parsing whole corpus 1:

```

word
|Poland|0|2|
|is|1|3|
word
|0|-1|-1|1|
|0|1|1|2|

```

A small piece of indexes and data base file for **words** after parsing whole corpus 1:

```

trigram
|pol|0|2|
|ola|1|2|
trigram
|0|-1|-1|2|
|0|1|1|2|

```

The usage of *regular expressions* [2] gives easy and fast method for moving through and modifying all four files. Regular expression are used in learning, checking and proposing.

3. CHECKING

After learning a checking function is proposed. In this case a *chunk* notion is proposed.

3.1. **Sentence checking.** Examples 2 and 3 can generate following chunks:

(5) *null* Poland is a small country
 correlation:1 root element correlation:1 correlation:1 correlation:1 correlation:1

(6) Poland is a small country · Poland is not
 correl.:1 correl.:1 correl.:1 correl.:1 root element correl.:1 correl.:1 correl.:1 correl.:1

Having normalized correlation values² reliability of each chunk can be evaluated. In the case 5 it is $\frac{5}{5} = 1$ and in the case 6 it is $\frac{8}{8} = 1$.

Now let's consider following input: "Poland is not a contry.". In this case there are 6 root elements so it give us 6 chunks. The output could look like:

```

* Poland is not a _
Trust: 0.8
* Poland is not a _ .
Trust: 0.83
_ Poland is not a _ . *
Trust: 0.71
_ Poland is not a _ . *
Trust: 0.71
- - - contry - -
Trust: 0.0
is not a _ . *
Trust: 0.8

```

Where "*" means *null* element and "-" means that there is no correlation between specific root and branch element. All 6 chunks give a simple cluster problem [5], [4]. Every chunk can be extended to the length of input + 2 *null* elements by filling rest of the places with *null* elements. After that we get following vectors:

²So far we set 1 when correlation > 0 and 0 otherwise.

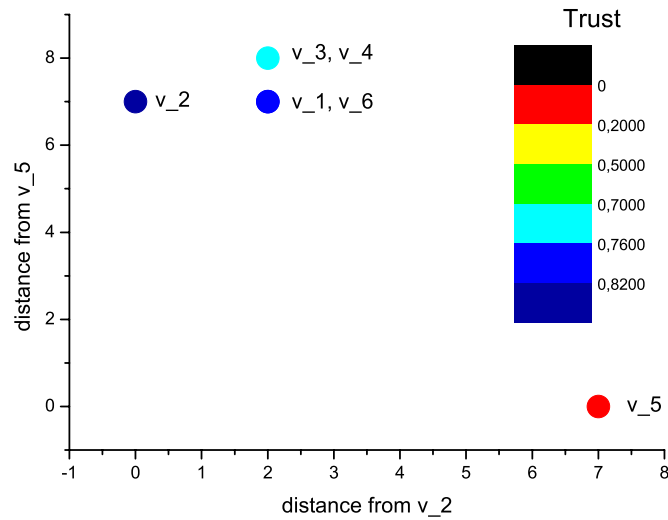


FIGURE 1. The clustering problem for word chunks.

$v_1 =$	*	Poland	is	not	a	-	*	*
$v_2 =$	*	Poland	is	not	a	-	.	*
$v_3 =$	-	Poland	is	not	a	-	.	*
$v_4 =$	-	Poland	is	not	a	-	.	*
$v_5 =$	*	-	-	-	-	contry	-	-
$v_6 =$	*	*	is	not	a	-	.	*

Natural metric³ is suggested he-

re: \sum_j if $v_i^j \neq v_k^j$ or $v_i^j = "-"$ or $v_k^j = "-"$ then 1 else 0. This metric and reliability function (figure 1) can help us to extract properly written sentences (section) or *generators* of properly written sentences (if there are any unknown correlations). In the case above we get two clusters and only one with non-zero mean reliability function: * Poland is not a _ . * | Trust: 0.76 . The generator is obtained by choosing the most frequent occurring element at every position⁴. This generator will serve in the next step for proposing the missing element.

3.2. Word checking. In the case of word checking the same method is applied but to the different elements. In example only "contry" word checking is showed in table 3. Same metric as before is used to find clusters. In this case 3 clusters are obtained (figure 2). Only one is with non-zero mean reliability function: * _ _ ntr try ryc * | Trust: 0.53 . This generator in the proposing faze can be use two-ways: to compare with proposition from the sentence generator or to propose a complete word and with that compose a new sentence.

³Proof should be added here.

⁴If some frequencies of a single element are the same then two or more generators are obtained. The mean trust value could be different.

$v_1 =$	-	con	-	-	-	-	*	*	Trust: 0.0
$v_2 =$	-	-	ont	-	-	-	-	*	Trust: 0.0
$v_3 =$	*	-	-	-	ntr	try	ryc	*	Trust: 0.5
$v_4 =$	*	-	-	-	ntr	try	ryc	*	Trust: 0.5
$v_5 =$	*	*	-	-	ntr	try	ryc	*	Trust: 0.6

TABLE 3. Normalized chunk vectors and reliability function values

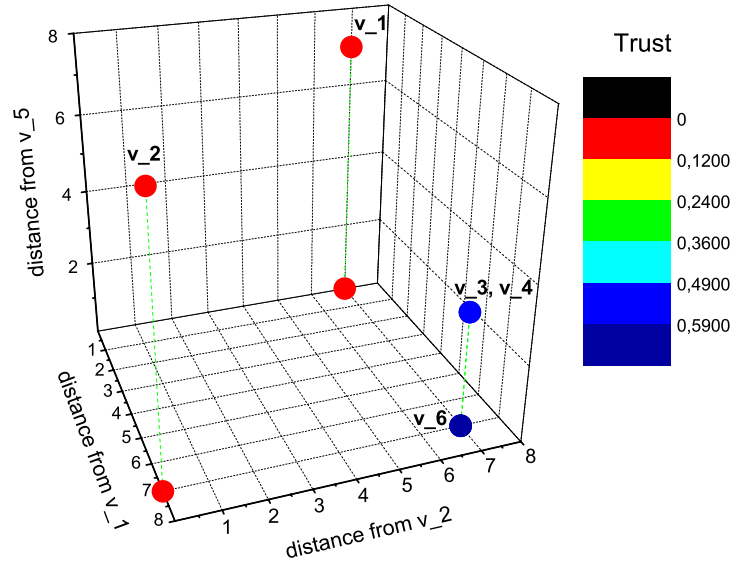


FIGURE 2. The clustering problem for tri-gram chunks.

4. PROPOSING

Third step in the recognition process is to propose a better sentence that the one badly written. At this stage an input sentence: "Poland is not a contry." gave one generator per each level (the section-sentences level and the word level): * Poland is not a _ . * and * _ _ _ ntr try ryc *.

4.1. Searching the solution. At the section-sentences level there are 5 tasks to be carried out: what correlates with "Poland" at the relative position 4, what correlates with "is" at the relative position 3 and so on. The data base with correlations between words is have to be searched. For that purpose regular expression are being used. When decoded following string are found:

```

|Poland|4|country|...|
|Poland|4|city|...|
...
|a|1|country|...|
|a|1|city|...|
|a|1|small|...|
...

```

In the case of corpus 1 only "a" correlates with more than two elements. Non the less only two elements are mutual for every root element: "country" and "city". Now comparison to the word-level generator can be made:

$v_1 =$	*	cou	oun	unt	ntr	try	ryc	*
$v_2 =$	*	cit	ity	tyc	*			
$v_3 =$	*	-	-	-	ntr	try	ryc	*

Because vector v_2 is of different dimension this solution can be discarded. If still the solution of this problem would not be clear filling the blank spaces of v_3 vector is possible with the same tasks-based method as at the sentences-section level:

```

|ntr|-1|unt|...|
|try|-2|unt|...|
|ryc|-3|unt|...|

```

Spaces are filled one by another in order to have more possible confirmation of choosing right tri-gram. In the case of v_3 vector there are 3+4+5 tasks to be carried out. If more than one combination of the fulfilling vector v_3 are possible then the nearest in sense of proposed metric is to be taken.

4.2. The solution. The solution in case of presented example is very clear: "**Poland is not a country**". More experiments should be conducted on larger training corpus and with more complicated badly written English sentences. So far it can be seen that Hecht-Nielsen method can be generalized and used in various way.

REFERENCES

1. Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer, *Class-based n-gram models of natural language*, Computational Linguistics **18** (1992), no. 4, 467–479.
2. Jeffrey E. F. Friedl, *Mastering regular expressions*, 2 ed., O'Reilly, 2002.
3. Robert Hecht-Nielsen, *A theory of cerebral cortex*, Technical Report 0301, University of California, San Diego, Institute for Neural Computation, 2003.
4. R. T. Ng and J. Han, *Efficient and effective clustering methods for spatial data mining*, 20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile proceedings (Los Altos, CA 94022, USA) (Jorgeesh Bocca, Matthias Jarke, and Carlo Zaniolo, eds.), Morgan Kaufmann Publishers, 1994, pp. 144–155.
5. E. Rasmussen, *Information retrieval: Data structures and algorithms*, ch. Clustering algorithms, Prentice Hall, 1992.

NICHOLAUS COPERNICUS UNIVERSITY, DEPARTMENT OF INFORMATICS
E-mail address: pawelm@phys.uni.torun.pl