TKK Reports in Information and Computer Science

Espoo 2010 TKK-ICS-R30

CONTEXTUALLY SELF-ORGANIZED MAPS OF CHINESE WORDS

Teuvo Kohonen

Aalto University
School of Science
and Technology

TKK Reports in Information and Computer Science

Espoo 2010 TKK-ICS-R30

CONTEXTUALLY SELF-ORGANIZED MAPS OF CHINESE WORDS

Teuvo Kohonen

Aalto University School of Science and Technology
Faculty of Information and Natural Sciences
Department of Information and Computer Science

Aalto-yliopiston teknillinen korkeakoulu
Informaatio- ja luonnontieteiden tiedekunta
Tietojenkäsittelytieteen laitos

Distribution:
Aalto University School of Science and Technology
Faculty of Information and Natural Sciences
Department of Information and Computer Science
PO Box 15400
FI-00076 AALTO
FINLAND
URL: http://ics.tkk.fi
Tel. +358 9 470 01
Fax +358 9 470 23369
E-mail: series@ics.tkk.fi

ISBN 978-952-60-3131-6 (Print)
ISBN 978-952-60-3132-3 (Online)
ISSN 1797-5034 (Print)
ISSN 1797-5042 (Online)
URL: http://lib.tkk.fi/Reports/2010/isbn9789526031323.pdf

Multiprint
Espoo 2010

Abstract: This is a technical report on the implementation of contextual SOMs of Chinese words. Several new developments are introduced. It seems that when the words are ordered on the SOM topologically, the order is not only determined by the word classes, but also the roles of the words as sentence constituents are reflected in their position on the SOM.

Keywords: Chinese text analysis, cognitive linguistics, contextual map, self-organizing map, semantic representation, SOM

CONTENTS

Contextually Self-Organized Maps of Chinese Words

Teuvo Kohonen

Abstract

This is a technical report on the implementation of contextual SOMs of Chinese words. Several new developments are introduced. It seems that when the words are ordered on the SOM topologically, the order is not only determined by the word classes, but also the roles of the words as sentence constituents are reflected in their position on the SOM.

1. Introduction

1.1 *Contextual maps*

The *self-organizing map (SOM) algorithm* (Kohonen, 2001) approximates a distribution of complex data items in an ordered fashion, using a *finite set of adaptive models* that have the same format as the items. Usually, a two-dimensionally ordered array of the models, called a *map* is developed in the learning process. The dissimilarities of the models are reflected in their geometric distances along the SOM array.

In *text analysis* one may be interested in the statistical regularities and similarities of *local contexts*, i.e., patterns formed of successive words (Ritter and Kohonen, 1989). In order to eliminate the effect of the word forms on the contextual structures, and to concentrate on the word patterns, i.e., local combinations of the words only, without paying attention to the writing forms of the words themselves, one ought to select representations for the words that are mutually as uncorrelated as possible. Thus, in order to eliminate the effect of the word forms, a random code may be assigned to represent each word. However, when using the basic SOM method for the contextual analysis, all of the input items ought to be *metric vectors*, and thus a natural selection for the random code would be a high-dimensional Euclidean vector with random elements, the typical dimensionality of which is a few hundred. In order that the coding would be independent of the directions of vectors, the random numbers ought to be normally distributed.

One particular problem encountered in context analysis is that the words in most languages are inflected, and the inflections carry semantic information, too. One naïve method is to treat every inflected form as a different word. Another method may be to reduce each word to its base form or word stem, whereby, however, the information connected with the inflections is lost. Nonetheless there also exist languages such as Chinese, where the words are not inflected at all, and which would then be ideal for the context experiments.

Among the applications of the self-organizing map, relatively few studies have been pursued on such *contextual maps* (also called the semantic maps or word maps). One reason for this may be that the contextual maps have only few practical applications, for instance, in text analysis and perhaps as feature extractors in textual data mining. Nonetheless the contextual maps have an important role in cognitive linguistics.

A typical contextual map results in the following type of a computing process. Consider a piece of natural text regarded as a string of symbols, each symbol standing for a word and having been chosen, e.g., as a unique high-dimensional Euclidean vector with randomly chosen elements. (For the simplicity of discussion we may regard the punctuation marks in sentences as "words," too.) A (local) *contextual feature* around a word is then defined to consist of m successive words. These words may be selected in different ways. One traditional choice is a *triplet* formed of three successive words, whereupon such a contextual feature is associated with the middle word, but experiments have also been carried out using only the previous and the next word around a word position to constitute the contextual features. In this work I use *quintuplets* of words as local features.

As we are usually interested in the *statistical features* of the text, we may form, e.g., for each word some kind of *averaged vector-valued feature* over the whole text. In averaging, the metric vectors as symbols for words come in handy. Consider that each symbol, corresponding to each different word w were represented by a unique Euclidean vector $\mathbf{r} = \mathbf{r}(w)$ that is an n-dimensional random vector having the (0,1)-normal distribution. Then the averaged contextual feature $\mathbf{x}(w)$ of word w may be defined as the $3n$-dimensional vector concatenated of three parts,

$$\mathbf{x}(w) \ = \ \text{avg}_{\,i}\,(\,[\mathbf{r}_{i-1} \quad \varepsilon\,\mathbf{r}_i \quad \mathbf{r}_{i+1}] \mid w)\,, \qquad (1)$$

where avg $_i$ (**.** | w) means the average over all positions i in the text, on the condition that the contextual feature relating to position i belongs to word w (i.e., on the condition that $\mathbf{r}_i$ is the random-vector representation of word w). The factor ε is a small scalar, and it defines the relative weight of the middle part with respect to the left and right parts in the feature. One may often select $\varepsilon = 0$. The expression (1) can be justified by the following two examples.

The set of expressions $\{\mathbf{x}(w)\}$ shall be used as inputs to an SOM for its training. In order to simplify the following explanation, assume tentatively in this subsection that the *dot-product SOM* were formed. In it, the matching of the input vector with the various model vectors (weight vectors) of the SOM is defined in terms of the dot product. In competitive training, the *winner model* is the one that has the highest dot product with the input vector. Every training step involves the updating of the winner model as well as of its topological neighbors on the SOM array.

Example 1. Consider two triplets $\mathbf{x}_1 = [\mathbf{a}\ \mathbf{c}\ \mathbf{b}]$ and $\mathbf{x}_2 = [\mathbf{a}\ \mathbf{d}\ \mathbf{b}]$, where **a**, **b**, **c**, and **d** are metric-vector representations of words. Let **c** and **d** correspond to *synonyms* or *antonyms*, such as 'big' and 'large', or 'big' and 'small' that may occur with a roughly similar probability in an extensive text corpus. Consider also that $\mathbf{x}_1$ and $\mathbf{x}_2$ occur as training inputs to an SOM and will be mapped onto it. Now, since **c** and **d** are uncorrelated, their similarity, measured by their dot product (**c**, **d**) is much smaller than (**a** ,**a**) or (**b**, **b**). In other words, the vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ match almost perfectly and will be mapped very close to each other on the SOM. (If the factor ε in eq.(1) were equal to zero, the matching between $\mathbf{x}_1$ and $\mathbf{x}_2$ would be perfect.) On the other hand, two triplets $\mathbf{x}_1$ = [**a c b**] and $\mathbf{x}_2$ = [**e d f**] with **e** and **f** different from **a** and **b** will be mapped far away from each other.

Example 2. Another specific set of inputs to the SOM is a subset

$$\{\mathbf{x}_i\} = \{[\mathbf{a}\ \mathbf{c}_i\ \mathbf{b}]\},\ i = 1,2, \ldots, m\ , \tag{2}$$

where the $\mathbf{c}_i$ are words, all of which occur in the text with roughly equal probabilities. An example of such words is a subset of *numerals*. However, since the different $\mathbf{c}_i$ are represented by different random vectors, their mutual dot products are small, whereupon the subset of inputs $\{\mathbf{x}_i\}$ will be mapped into a narrow cluster on the SOM.

1.2 *Early history of this work*

Contextual SOMs have been constructed since 1989. To the knowledge of this author, all of them have been based on English texts, whereupon the words have usually been defined exactly as they appeared in the text, i.e., various inflected forms of the same word were always regarded as different words. It occurred to this author already around 1994 that since the Chinese words are not inflected, the contextual maps of Chinese texts might be produced more consequently than in many other languages. However, a major handicap of the automatic computerized processing of Chinese texts is that the words are not separated by spaces, but all of the *characters* ("letters") of which the Chinese words are formed are coalesced within the same sentence into a contiguous string of characters. In 1994 we were not aware of any Chinese text corpora segmented into words.

As indicated above, the Chinese words are formed of one or more "letters" called the *characters*. There exist roughly 2000 characters nowadays. Words formed of one character constitute roughly about 63 per cent of all words. Two-character words make up about 34 per cent, three-character words about 2 per cent, and four-character words about 1 per cent of the text, respectively. (These figures are from the extensive MCRC corpus used in this work.) Words consisting of five or more characters amount to less than one in a thousand words.

My first idea was to make a map of the Chinese *characters* only, by forming the context triplets out of successive characters of the given text. As the first results did not seem quite satisfactory, I got another idea of defining the middle part in the triplets of eq.(1) alternatively out of *one or two successive characters*, whether they represented real words or not, and using *single characters* bordering to the middle part on the left and right as the other constituents that make up the local context. In this way I hoped that the correlations between true words would be reflected stronger in this experiment. In order that both single characters and double characters would be involved in the same experiment as training data, the training vectors for the SOM had to be produced *in two* *separate passes* of the text. In the first pass, all of the triplets of the successive characters were recorded, and the first set of input vectors **x**(w) was constructed as explained above. In the second pass, all successive *quadruplets* of the text were recorded, and the middle part of eq.(1) was formed of the middlemost two characters, whereas the first and fourth character in the quadruplets were used for the rest of the context.

The subsets of input vectors **x**(w) obtained in these two passes were merged and used for the training of the SOM. We obtained the source codes of the Chinese texts in electronic form from Chinese newsletters sent to Chinese students, who were studying abroad. These texts were kindly made available to us by Miss Li Liu, who studied in our university at that time. Every newsletter started with a list of characters used in it, together with their alphanumerical codes (not the same as the Unicodes used nowadays), so the conversion of the data into the form of eq.(1), and preparation of the Chinese characters for the labeling of the SOM were a rather straightforward task. The program codes were written by Dr. Jari Kangas in 1994. Of course, no genuine word maps could be obtained in this way. However, I have to remind that around 1994 we did not have available any automatic segmentation methods that would have partitioned the Chinese texts into words, and manual segmentations were neither available to us. Nonetheless, since approximately 96 per cent of the Chinese words are formed of either one or two characters, there is already a rather strong correlation of the successive single and double characters to the true Chinese words, whereby some positive results were already obtained in the above experiments. In the series of images shown in Fig. 1 we see parts of the SOM thereby produced, with English translations written at the side of the Chinese characters or pairs of characters in the case that they represented real words.

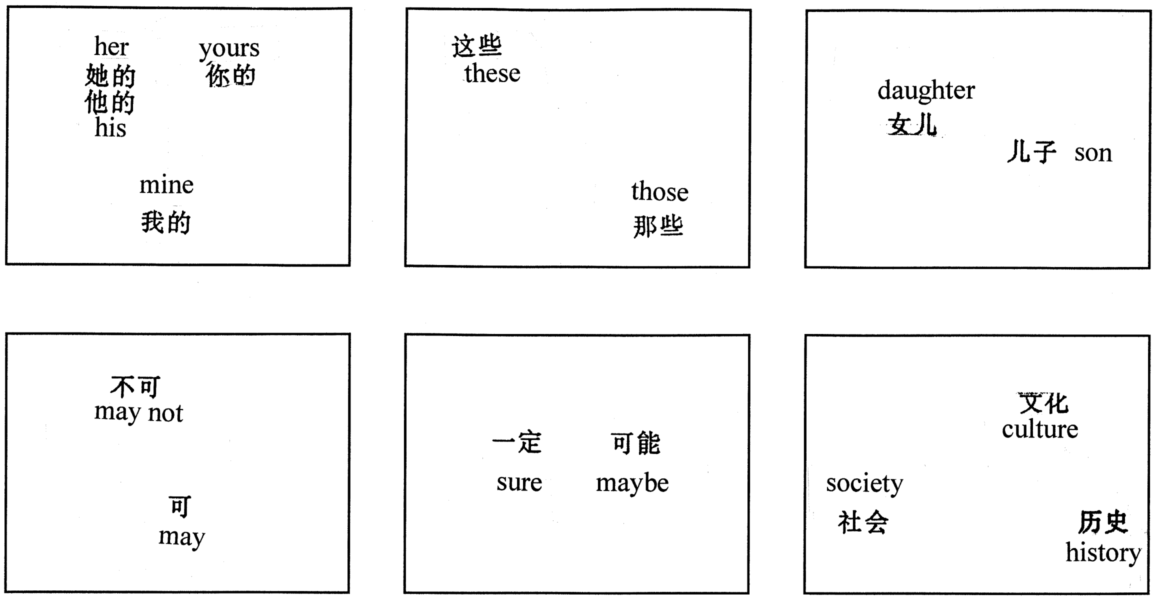

Fig. 1. Parts of a 750 mm by 600 mm wide map of Chinese characters

It turned out that local clusters of the type described in Examples 1 and 2 were indeed formed. However, no global ordering of the SOM was yet visible, which is quite understandable due to the obvious defects of the naïve method.

2. The present MCRC experiment

2.1 Start of cooperation

At the WSOM 2009 (Workshop on Self-Organizing Maps) held in St. Antonio, Florida, U.S.A., I met Professor Ping Li of Pennsylvania University, and told about my early attempts to produce Chinese contextual maps. We both realized that it would be very important to start developing genuine word-based contextual maps for the Chinese language, in order to see whether they would differ from those constructed for English. In fifteen years, rather effective automatic segmentation methods for the Chinese language had been developed, and the construction of Chinese contextual maps seemed possible. What was perhaps even more intriguing, however, was that big Chinese text corpora had been segmented manually and even parsed linguistically during that time. One of such corpora is the Beijing Normal University Corpus (Shu et al., 2003), which is an electronic, lexically segmented database that contains all of the texts from elementary-school textbooks used in Beijing. Another corpus, called the MCRC (Modern Chinese Research Corpus) (Sun el al., 1996), is an electronic collection of text material from newspapers, novels, magazines, TV shows, folktales, and other text material from modern Chinese media. The MCRC contains about 1’500’000 words provided with linguistic classification of the words by Dr. Hongbin Xin. I received the MCRC corpus in the fall of 2009, and with the help of Drs. Zhirong Yang and Timo Honkela of our university, this material, which was sent to us in Unicode form, was converted into decimal tables (in order that I could compute the input vectors according to eq.(1) and process them using MATLAB scripts, as I shall explain below in more detail).

2.2 New developments of the contextual-map method

As I had practically no knowledge in the Chinese language, I decided to calibrate the contextual SOMs (that could be constructed automatically) according to given word classes, not by the individual words, as normally done. This became possible to me because every word in the corpus was labeled according to some of the 89 linguistic classes selected.

Also, in order to carry out a number of variable experiments, I decided to use the MATLAB scripts for writing the program codes. An extra incentive for this choice was that an extensive MATLAB software package called the SOM Toolbox, with flexible graphic functions, had been developed in our laboratory several years earlier (Vesanto et al., 1999). So it was quite possible for me to write the scripts and to carry out these experiments by myself.

2.2.1 *Hash coding of the vocabulary*

I decided to use the whole MCRC corpus for the construction of a SOM, not only any subset of the most frequent words. At several stages of computation, I therefore needed a lot of table look-ups. The MATLAB does not have any effective function for the processing of symbol strings. I decided to organize the word vocabulary by the *hash coding* method, using the Unicode representations of the Chinese characters to represent the words, and treating each Unicode as a decimal number.

Many different hash-coding schemes were developed at the time when the computer memories still had limited capacities (cf. e.g., Kohonen 1980, Chapter 2). After that, a vast increase in the memory capacities has taken place. Although in this study the available memory space was finally limited by the MATLAB system, it was anyway orders of magnitude larger than that of the old memories. Therefore, one of the simplest but anyway fast hash table organizations, the *open-addressing hash table*, could be implemented easily. One additional aspect useful in this work was that for an *ad hoc* database like the present vocabulary, where the contents of the memory are known beforehand, one can choose the hash table size and the hashing function in such a way that a limit, say, three, can be set to the number of reserve locations needed, and still a perfect hash table can be constructed (i.e., every stored item can be found by a maximum of four memory accesses). In the MCRC application studied in this work, the size of the vocabulary was 48’191 words, and the hash table was made to contain 497’198 addresses. With the chosen details of the hash table as explained below in more detail, the average number of memory accesses per searching operation was approximately 1.14.

The relative number of Chinese words represented by more than four characters, at least in the MCRC corpus, is less than 1/1000 of all words, and if such words are neglected in the training of a SOM, the error thereby committed is negligible, regarded as “noise.” Therefore, in my experiments, only words consisting of up to four characters were stored in the hash table. At each address of the latter there are six fields, four of which can represent up to four characters of a word, one field is needed for the indicator of a collision called the *flag* as explained below, and one field is assigned to indicate the ordinal number (“index”) of the word in the vocabulary, respectively. The Unicode of a one-character word will be stored in the first field of the memory location, and the three following fields shall contain the value of zero. For two-character words, the two Unicodes are stored in the first two fields of the memory location, and the next two fields contain the zero value, and so on.

When writing items into the hash table, first the so-called *hash address* of each given word is computed by the *hashing function*, a pseudorandom arithmetic function of the word, and a trial is always made first to store the word at the hash address. In this study the hashing function was defined as the following. Let the numerical Unicode values of the characters of a word be c_1, c_2, c_3, and c_4. For one-character words we shall take $c_2 = c_3 = c_4 = 0$. The two-character words shall have $c_3 = c_4 = 0$, and the three-character words shall have $c_4 = 0$. Let b_1, b_2, b_3, and b_4 represent fixed parameter values. The hashing function that defines the hash address is defined as

$$h = \mathrm{mod}\,(b_1\, c_1 + b_2\, c_2 + b_3\, c_3 + b_4\, c_4, T) + 1\ , \qquad (3)$$

where $T = 497'148$ is the number of addresses in the hash table.

All of the flags have initially the value of zero. The first entry (word) is stored at the address h, and the corresponding flag in the fifth field at the corresponding memory address is set to the value of 1, showing that this address will be occupied from this time on.

When storing further entries in the above manner, at first the probability for finding an unoccupied hash address is relatively large, close to unity, whereas with time this probability is decreased to about .86 in our application. Nonetheless, at any time when hitting an *occupied* hash address (with the flag value 1), the next attempt is to store the new entry at the first *reserve location* defined as

$$h' \ = \ \mathrm{mod}\,(h\,, T) + 1\ . \qquad (4)$$

If its flag is still equal to 0, the new entry is stored there, and the flag is set to 1. However, sooner or later, when adding new entries, we will find that both the hash address and its first reserve location will be occupied, whereupon an attempt must be made to store the entry at the second reserve location defined as

$$h'' \ = \ \mathrm{mod}\,(h'\,, T) + 1\ . \qquad (5)$$

If this place is occupied, too, we must use the third reserve location in analogy with eqs. (4) and (5) to store the entry.

In the traditional open-addressing hash coding method, the chain of reserve locations must be allowed to grow until the whole hash table is filled up. However, as mentioned above, the present application is a special one, because the contents of the hash table are fixed and known beforehand. Then, for instance for a vocabulary of 48'191 words and taking $T = 497'198$, and using the parameter values $b_1 = 250'000$, $b_2 = 250$, $b_3 = 25$, and $b_4 = 1$, one indeed obtains a hash table where no more than three reserve locations are needed, as can be verified numerically. This fact makes the structure and programming of the hash table very simple and its operation very fast.

When the whole vocabulary has been organized as a hash table, a quick reading of the ordinal number of a given word in the vocabulary, and other eventual information associated with it (like quick definition of the random-vector representation of a word as will be explained below) will take place in the following operations: 1. Compute the hash address for the given search argument word (c_1, c_2, c_3, c_4) according to eq.(3). If the word stored at the hash address is identical with the search argument word, read out the associated information (e.g., the ordinal number of the word). 2. However, if the stored word and the search argument disagree, compute the address of the first reserve location according to eq.(4). If the word found now agrees with the search argument, read out the associated information. 3. In the case that there is still a disagreement, find the address of the second reserve location according to eq.(5), and so on.

2.2.2 *Construction of the random-vector inputs to the SOM*

The MCRC corpus is so large that it would be impossible to store the input file of the SOM in the working memory. Consider that ten bytes are needed for each numerical variable in MATLAB. If the dimensionality of the input vectors **x** were equal to 500, and if the number of words in the MCRC corpus were 1’500’000, we would need 7.5 gigabytes of memory capacity for this file only. My solution to avoid storing a large data file was to compute the input vectors $\mathbf{x}(w)$ *during the execution of the training program*. This is possible, if before the computation of each random vector $\mathbf{r} = \mathbf{r}(w)$, the random-number generator is initialized by the ordinal number of the word w in the original vocabulary, and this ordinal number, as a function of w, is obtained directly from the hash table. This procedure will also bring about an extra advantage, as shall be explained in the next subsection, namely, that the dimensionalities of $\mathbf{r}_{i-1},$ $\mathbf{r}_{i,}$ and $\mathbf{r}_{i+1}$ can be selected *separately and differently* for each of them, which will make it possible to use the memory capacity more effectively and to reduce the computing time.

2.2.3 *Computation of the SOM*

The size of the SOM array was selected as rather small, 40 by 50, in order to save memory but still to be able to discern the cluster structures of the word classes on it. However, the structure and dimensionality of the input vectors was determined on the basis of earlier experiments. It turned out that in my laptop computer, on which I carried out all of the experiments, the maximal dimensionality of the input vectors **x** (and that of the model vectors) in this problem could not be essentially higher than 650. After a number of experiments I decided to use a context that consisted of *five* successive words, represented by the random vectors $\mathbf{r}_{i-2}.$ $\mathbf{r}_{i-1},$ $\mathbf{r}_i$, $\mathbf{r}_{i+1,}$ and $\mathbf{r}_{i+2}$, respectively. The factor ε in eq.(1) was taken equal to 1, but the dimensionality of the middle vector $\mathbf{r}_i$ was then selected correspondingly smaller, equal to 50. The dimensionalities of both $\mathbf{r}_{i-1}$ and $\mathbf{r}_{i+1}$ were 200, and those of $\mathbf{r}_{i-2}$ and $\mathbf{r}_{i+2}$ were equal to 100, respectively. In a number of earlier experiments it had turned out that the results are slightly improved if the contexts are wider than a triplet, but there is essentially no advantage of using wider contexts than word quintuplets. Here $\mathbf{r}_{i-1}$ and $\mathbf{r}_{i+1}$ have the highest weight in determining the dot product between the input vector and the model vector.

It is to be noted that if the dimensionalities of the **r** vectors are selected individually as above, a word cannot have a unique random-vector representation any longer. However, notice that the **r** vectors are now generated during the training of the SOM, so the random vector that represents a word now depends on the relative position of the word with respect to the middle word $\mathbf{r}_i$. The different dimensionalities will not cause any problem in the matching of the input vector with the model vectors, because the representations of the words of the input vector and those of the model vector always consequently refer to the same relative position where the dimensionalities are the same.

In order to resort to well-documented SOM software, I decided to apply the SOM Toolbox program package (Vesanto et al., 1999), which is written as MATLAB scripts. The linear initialization of the model vectors was used. I also used the batch training algorithm with a coarse training phase followed by a fine tuning phase. The training vectors **x** were normalized to unit length. This does not yet mean that the model vectors would become exactly normalized, too, but the deviations of their converged values from the unit length can be shown to be very small, and at least do not affect the topological ordering of the model vectors. When using the normalized input vectors, on the other hand, the calibration of the SOM (finding the clustering of the word classes on it), however, could be based on the dot-product matching, and to that end the model vectors too were normalized after training.   .

In a recent study (Kohonen, 2009) I found out that if the neighborhood function applied during the fine-tuning phase of learning does not change with time, the SOM algorithm will converge in a finite number of training cycles. This property was utilized in the present work to guarantee that the SOM had really converged.

2.2.4 *Hit diagrams on the SOM*

As indicated above, the primary goal in this study was to visualize the relations between word classes by the SOM.

When constructing the SOM, averages of the context vectors **x** relating to *each unique word in the text* were used as training data. In order to be consequent, the calibration of the SOM must also be based on similarly defined input vectors, taking only subsets of input vectors relating to particular words, word classes, or other subsets of words into account.

Also, when locating the best match of a particular word w on the SOM, the average context vector $x(w)$ relating to it must be used as input.

For a subset of words such as all verbs, all adjectives, all nouns, etc., one may construct its *hit diagram*, i.e., the number of matches of this subset of words with the various SOM locations. The number of matches (hits) at a particular SOM location is indicated by gray levels.

Verbs. Fig. 2 shows the distribution of all *verbs* on the SOM. At first sight their distribution may seem rather fragmented and uneven. However, the set of verbs probably consists of several subsets that have their own distributions. It may also be necessary to point out that there is no principal reason for the contexts being clustered just according to the major *word classes*. It may become more obvious from the continuation that it may be *the role of the words as sentence constituents* that might be better reflected in the word order, and thus in the local context. After the discussion of every major word class I shall present examples of particular word classes that have a cluster in a particular location on the SOM, where the major word class also has a cluster, and this suggests what kind of roles the subsets of words of the major class may have in these areas. For instance, close to the upper left corner of the hit diagram of the verbs there is a faint cluster that probably represents *predicates,* as deducible when compared with the corresponding cluster of *predicative idioms*, shown in Fig. 3.

There exists an intense and rather round cluster in the middle, which does not correlate with the clusters of other words, and might describe, e.g., simple verbs without objects.

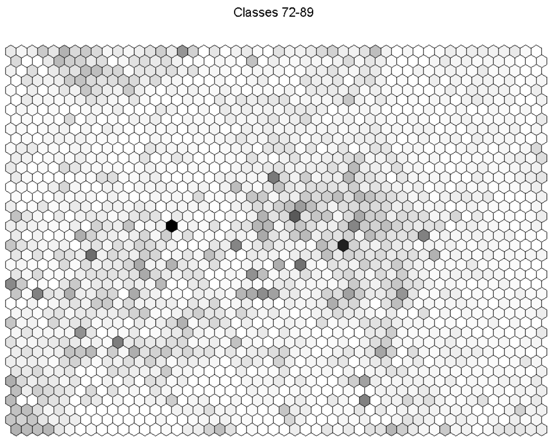

Fig. 2.. The hit diagram of all verbs on the SOM.

Predicative idioms. The Chinese language has a marked word class of predicative idioms. Fig. 3 shows that they are clustered at the top of the SOM, to the left of the middle, where many other word classes also have a cluster. It is to be believed that in this area, the clusters in the other word classes also have a predicative nature.

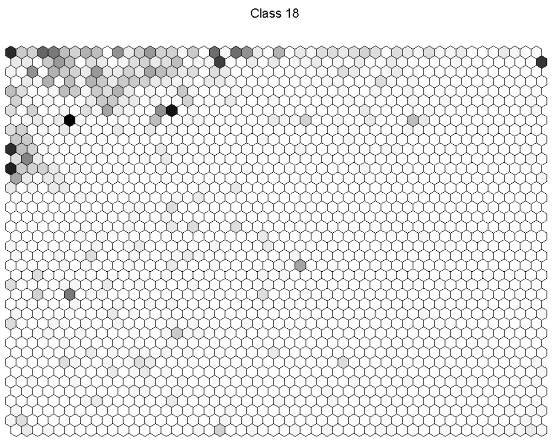

Fig. 3. The hit diagram of predicative idioms.

Nouns. Fig. 4 is the hit diagram of all *nouns*. It is essentially a broad belt surrounding the distribution of the verbs, but one can also discern certain interesting details in it. First, the region in the middle of the SOM, where the verbs have a big round cluster, is quite empty. So at least in this region the verbs and the nouns are segregated. Close to the upper left corner there is a weak cluster that almost coincides with that of the predicative idioms. It seems plausible that this cluster represents nouns related to the predicatives.

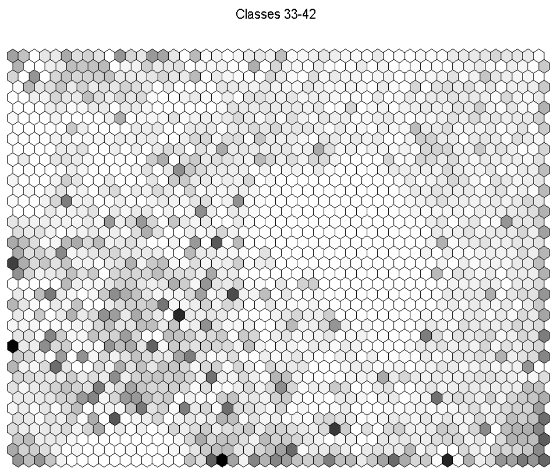

Fig. 4. The hit diagram of all nouns.

Somewhat to the left and down from the middle there is a broad area where different kinds of *names* are mapped. This can be seen from Fig. 5. One ought to notice that the MATLAB graphics automatically scales the gray levels, so due to some very strong points, the shade of the rest of the names in the cluster has remained rather pale.

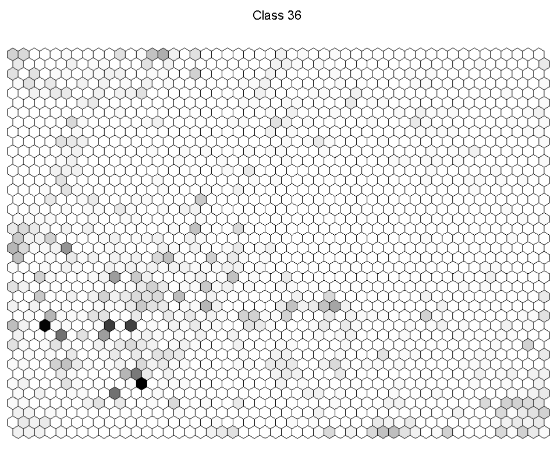

Fig. 5. The hit diagram of names.

Adjectives. In Fig. 6 we have the hit diagram of all *adjectives*. There seem to exist three main clusters in it. (Cf. also Fig. 14.)

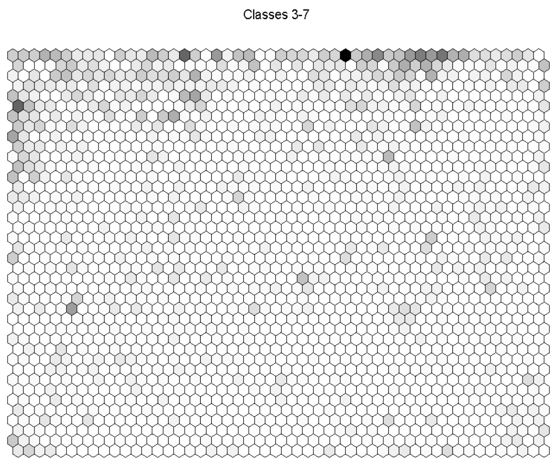

Fig. 6. The hit diagram of all adjectives.

At the top, close to the upper left corner, we have a broad cluster in the place where the predicates (verbs) and predicative idioms also have it (cf. Figs. 2 and 3). This cluster probably describes *predicative adjectives*.

At the top, in the right half of the map there is another more intensive cluster that seems to represent *attributes*. This conclusion follows from the observation that the *attributive pronouns* also have a cluster in the same place (cf, Fig. 7).

The third cluster in the hit diagram of the adjectives is discernible at the left side, rather close to the upper corner. This is the place where the *adverbial idioms* (cf. Fig. 8) also have a cluster. The adjectives of this cluster may have an adverbial character.

Attributive pronouns. The hit diagram of the attributive pronouns has a clear intense cluster shown in Fig. 7. Upon comparison with the adjective map in Fig. 6, the same cluster can indeed also be found among the adjectives.

Adverbial idioms. The cluster of adverbial idioms is visible in Fig. 8. One may compare it with the left side of the adjective map in Fig. 6.

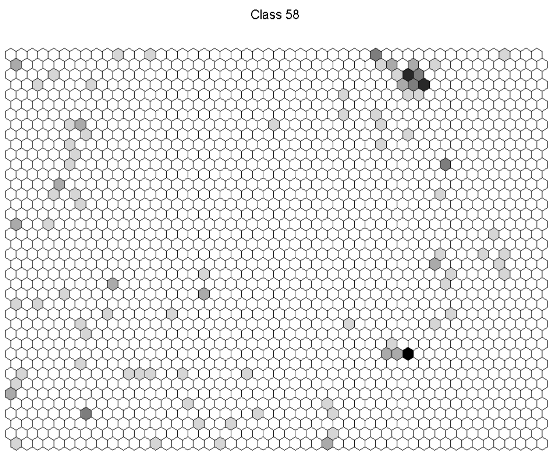

Fig. 7. The hit diagram of the attributive pronouns.

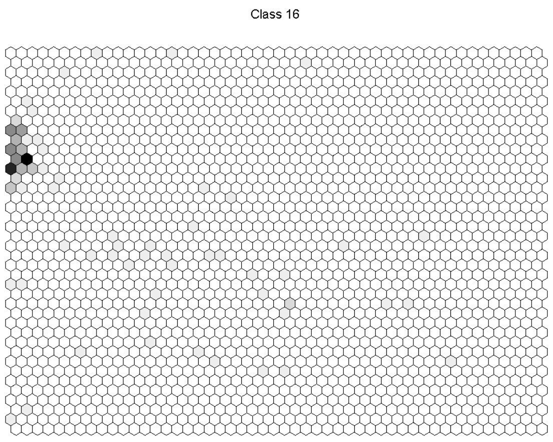

Fig. 8. The hit diagram of the adverbial idioms.

Numerals. In my experience, whatever versions of the contextual SOMs we have produced, the distribution of the *numerals* has always been very well clustered. This already came out in our naïve experiments around 1994.

Fig. 9 is the hit diagram of numerals obtained in my present experiments. It seems to be otherwise quite compact, except that the mapping of the numeral "zero" (Fig. 10) is quite separate from the rest, which is understandable on account of its special role in texts.

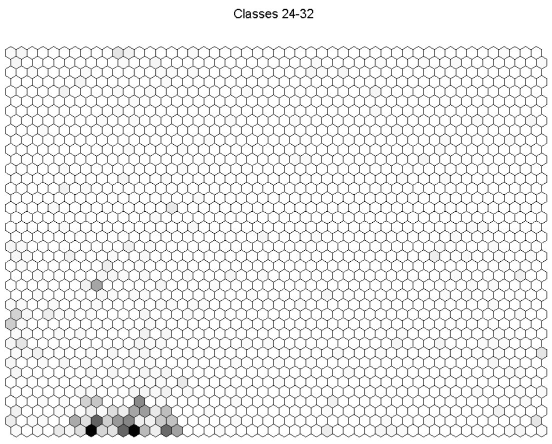

Fig. 9. The hit diagram of the numerals.

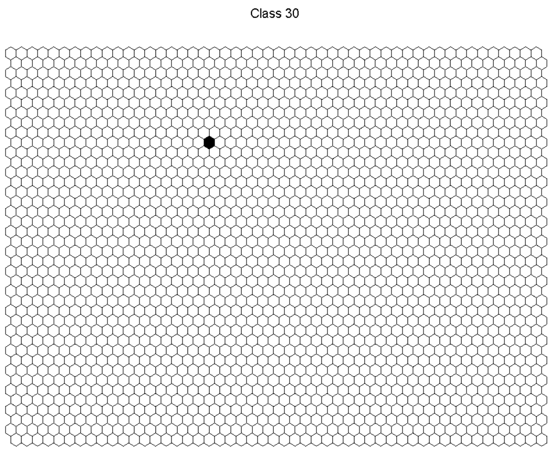

Fig. 10. The hit diagram of the numeral "zero."

Adverbs. The adverbs are related to the verbs ("ad-verb"), and on the SOM they are located at the fringes of the verb region, see Fig. 11. The main clusters surround the main region of the verbs.

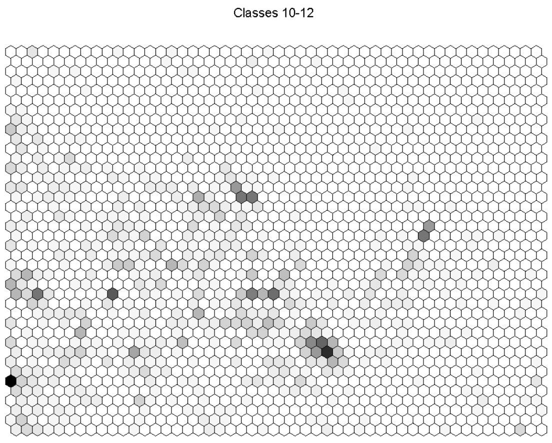

Fig. 11. The hit diagram of all adverbs.

Prepositions. The prepositions may be attached to any nouns, and therefore they are almost uncorrelated with the latter. However, since they occur close to the nouns in the texts, their contexts are mapped into the vicinity of the region of the nouns. Fig. 12 gives the distribution of all prepositions. In particular, the preposition “in” is mapped into a single location shown in Fig. 13, in spite of it occurring in many different contexts.

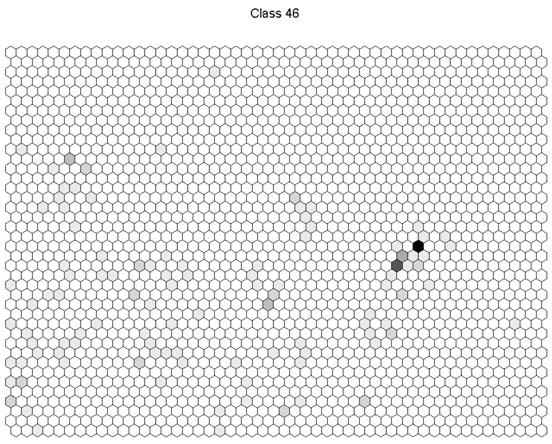

Fig. 12. The hit diagram of all prepositions.

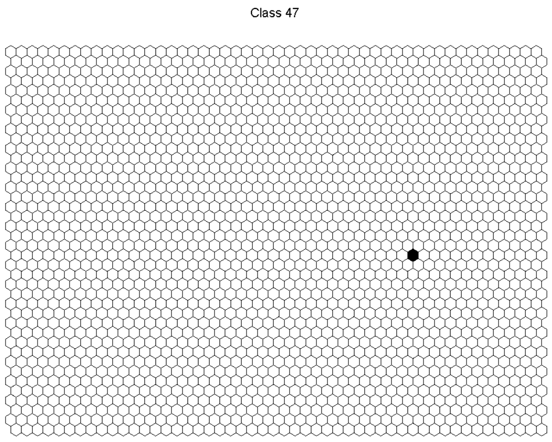

Fig. 13. The hit diagram of the preposition "in".

2.2.5 *Coloring of the zones of the most probable main word classes on the SOM*

When the SOM models were labeled according to the classification of the majority of classes of hits on them, we obtained the Bayes-type decision zones of the models, shown in Fig. 14. Here red means verbs, yellow adjectives, green nouns, blue adverbs, and white numerals, respectively. One can see, e.g., the three subsets of adjectives very clearly.

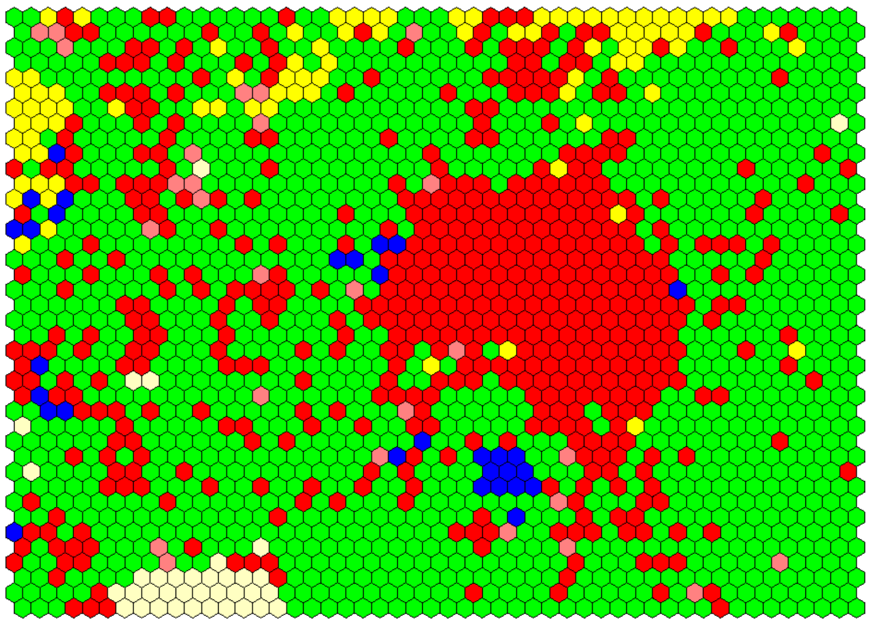

Fig. 14. The decision zones of the main word classes.

3. Discussion

It is a common feature of all self-organizing maps that their appearance can vary, depending on the dimensions of the array and the vectors, and the relative scaling of the input data. This is the same effect as when a solid body in a three-dimensional coordinate system is viewed from different angles. In the SOM experiments we usually have a very large number of dimensions. These differences may not be regarded as errors, as long as the main topological relations are preserved in the two-dimensional projection. I have repeated the experiments many times, using different parameters and different choices of the random vectors, in order to become convinced that the results are essentially right.

Also the degree of clustering of different word classes can be seen to vary. This, however, is probably due to the inherent nature of the data used in this experiment. If the texts were simple or literary, or would be written by the same authors, also the self-organizing maps would look simpler and more consequent. One cannot expect that the texts from the very different sources, and written by the numerous authors of the MCRC corpus would have consequently had the same style and similar usage of the word order. One should also pay attention to the fact that the present experiment was based on all words that occurred in the corpus, not only on a subset of the most frequent words.

References

1. T. Kohonen, *Self-Organizing Maps*, 3rd edition (Springer-Verlag, Berlin-Heidelberg, 2001)
2. H. Ritter and T. Kohonen, “Self-organizing semantic maps,” *Biol.Cyb*., 61, 241-254, 1989.
3. H. Shu, X. Chen, R. Anderson, N. Wu and Y. Xuan, ”Properties of school Chinese: implications for learning to read,” *Child Development*, 74, 27-47, 2003.
4. H. L. Sun, D. J. Sun, J. P. Huang, D. J, Li, and H. B. Xing, “Corpus for modern Chinese research,” in *Studies in the Chinese language and characters in the era of computers*, ed. by Z. S. Luo and Y. L. Yuan, pp. 283-294 (Tsinghua University Press, Beijing, China, 1996)
5. E. Alhoniemi, J. Vesanto, E. Alhoniemi, J. Himberg, K. Kiviluoto, and and J. Parviainen, “Self-organizing map for data mining in Matlab: the SOM Toolbox.,” *Simulation News Europe*, 25, 54, 1999. The SOM Toolbox software package is downloadable from www.cis.hut.fi/research
6. T. Kohonen, *Content- Addressable Memories* (Springer-Verlag, Berlin- Heidelberg, 1980)
7. T. Kohonen, I.T. Nieminen, and T. Honkela, ” On the quantization error in SOM vs. VQ: a critical and systematic study,” in *Advances in Self-Organizing Maps*, ed. by J. C. Principe and R. Miikkulainen, LNCS 5629 (Springer-Verlag, Berlin-Heidelberg, 2009)

(Submitted on April 7, 2010)

ISBN 978-952-60-3131-6 (Print)
ISBN 978-952-60-3132-3 (Online)
ISSN 1797-5034 (Print)
ISSN 1797-5042 (Online)