Pattern discovery in annotated dialogues using dynamic programming

Thierry Lecroq

Université de Rouen, LITIS EA 4108
76821 Mont-Saint-Aignan Cedex, France
Email: Thierry.Lecroq@univ-rouen.fr

Alexandre Pauchet

INSA-Rouen, LITIS EA 4108
BP 08 - 76801 Saint-Étienne-du-Rouvray, France
Email: Alexandre.Pauchet@insa-rouen.fr

Émilie Chanoni

Université de Rouen, Psy.NCA EA 4306
76821 Mont-Saint-Aignan Cedex, France
Email: Emilie.Chanoni@univ-rouen.fr

Gerardo Ayala Solano

INSA-Rouen, LITIS EA 4108
BP 08 - 76801 Saint-Étienne-du-Rouvray, France
Email: gerardo.ayala-solano@insa-rouen.fr

Abstract: This study describes a new heuristic to discover behavioural patterns in annotated dialogues: firstly, dialogues are transcribed and annotated; then, two-dimensional patterns are extracted; finally, the semantic pertinence of the extracted patterns can be evaluated by an expert. The dialogues are annotated by using a grid designed by the expert. Recurrent patterns are extracted in the annotated dialogues using dynamic programming with the help of a substitution matrix specifically designed for the task. This article focuses on the method developed for extracting the patterns and presents as application some extracted patterns on annotations of dialogues between parents and children during the narration of child stories.

Keywords: Pattern discovery, dynamic programming, semi-automatic dialogue analysis

1 Introduction

The design of computer systems that interact efficiently with users necessitates robust models. These models have not to be defined arbitrarily by the system designer but they have to be based on the study of human capabilities of interaction and communication. For instance, a friendly dialogic interface in natural language has to be efficient concerning the task model and has to integrate a communication behaviour as natural as possible. Among the various existing interaction models, the most common approach adopted is to study a corpus of annotated dialogues, traces or logs, in order to extract a set of significant recurrent behaviours. When the annotations are judiciously chosen, the recurrent behaviours appear as patterns repeated in the annotations. These regularities are often represented with automata [22], timed automata [18], Petri nets [16], sequence diagrams [20], and so on. All these representations use linear patterns.

In the particular case of corpus of complex dialogues, such as multimodal and/or affective dialogues, each utterance is annotated with a series of codes, following what is called a *coding scheme*. For instance, DIT++ [9] and the Emotion grid [10, 11] are good examples of multidimensional coding schemes. Therefore, extracting two-dimensional patterns in such annotated dialogues requires an approach different from the classic one.

The method presented in this article answer to this problematic thanks to the following scenario:

1. Transcription and annotation of the dialogues;
2. Extraction of alignments of two-dimensional patterns using dynamic programming;
3. Evaluation of the semantic pertinence of the extracted patterns.

To solve point 2, we first look at the situation that arises in one dimension. Strings (sequences of symbols belonging to an alphabet Σ) can easily be compared using dynamic programming. Applications of this problem include computational biology, computation linguistics or speech recognition. Trees can also be compared using similar techniques. The reader is referred to [19] for further details. A great number of works consider exact and fuzzy pattern (or approximate pattern) matching in two dimensions: given a two-dimensional pattern X and a two-dimensional text Y, find the occurrences of X in Y (see [2, 5, 6, 8, 14] for instance). Very little attention has been given to the alignments of two-dimensional matrices. Applications of this problem include fundamental topics such as data mining (in time series databases for instance) and image analysis. Krithivasan and Sitalakshmi [15] consider 2D patterns of exactly the same size while Baeza-Yates [7] considers only global alignments of 2D patterns.

There have been some efforts for indexing matrices using either suffix trees or suffix arrays (see [17] for a recent reference) but those indexes enable to find exact repeats and the generalization for finding approximate repeats is far from immediate. Recently, Arslan [4] considered the largest common subsequence of two patterns in dimension $d \geq 2$ but using different techniques. In particular, he only

used equality/inequality operations and cannot distinguished between low similar areas and completely different areas. In [3], the authors proved that finding the longest common substructures of two matrices in $\mathcal{NP}$-hard.

The edit distance between two matrices of dimension higher than one is difficult because there is no commonly accepted set of operations for transforming one matrix into the other. This article presents a new heuristic for comparing two matrices, more precisely it gives the recurrence formulas for performing global as well as local alignments of two-dimensional patterns of respective size M and N in $O(M \times N)$ space and time. For that, we need to precompute the similarities between all the prefixes of all the lines of the two patterns and between all the prefixes of all the columns of the two patterns. This can also be done in $O(M \times N)$ space and time.

The method is then used to detect patterns in annotated dialogues between parents and children during the narration of child stories. The process uses a substitution matrix, a vector of insertion costs and a vector of deletion costs, specifically designed by an expert for this task. This enables us to detect significant patterns common to several dialogues.

The remaining of this article is organized as follows: Section 2 recalls the dynamic programming techniques to align strings. Section 3 presents our new dynamic programming method to align two-dimensional patterns. Section 4 explains the evaluation of the method on annotated dialogues between parents and children. Finally, Section 5 concludes and presents some future work.

2 Aligning strings

Alignments are usually used to compare strings. They are widely used in computational molecular biology. They constitute a mean to visualize resemblance between strings. They are based on notions of distance or similarity. Their computation is usually done by dynamic programming. We can consider two different kinds of alignment of two strings x and y: global alignment (that considers the whole strings x and y), and local alignment (that enables to find the segment of x that is closer to a segment of y).

2.1 Global alignment

A global alignment of two strings x and y can be obtained by computing the distance or the similarity between x and y. The notion of distance between two strings is widely used to compare files. The **diff** command of UNIX operating system implements an algorithm based on this notion, in which lines of the files are considered as symbols. The output of a comparison made by **diff** gives the minimum number of operations (substitute a symbol, insert a symbol, or delete a symbol) to transform one file into another. This command performs a global alignment between the two files.

Let us define the edit distance between two strings x and y as the minimum cost of elementary edit operations that enable to transform x into y. The elementary edit operations are: the substitution of a symbol of x at a given position by a symbol of y; the deletion of a symbol of x at a given position; the insertion of a symbol of y in x at a given position.

Example:

A	**C**	**G**	–	–	**A**
A	**T**	**G**	**C**	**T**	**A**

is a global alignment of **ACGA** and **ATGCTA**.

A solution can also be given as an edit script as follows:

Operation	Resulting string
substitution of **A** by **A**	**A**
substitution of **C** by **T**	**AT**
substitution of **G** by **G**	**ATG**
insertion of **C**	**ATGC**
insertion of **T**	**ATGCT**
substitution of **A** by **A**	**ATGCTA**

A cost is associated to each elementary edit operation. For $a, b \in \Sigma$: $Sub(a, b)$ denotes the cost of the substitution of the symbol a by the symbol b, $Del(a)$ denotes the cost of the deletion of the symbol a, $Ins(a)$ denotes the cost of the insertion of the symbol a. This means that the costs of the edit operations are independent of the positions where the operations occur. We can now define the edit distance of two strings x and y by $d(x, y) = \min\{\text{cost of } \gamma \mid \gamma \in \Gamma_{x,y}\}$ where $\Gamma_{x,y}$ is the set of all the sequences of edit operations that transform x into y, and the cost of an element $\gamma \in \Gamma_{x,y}$ is the sum of the costs of its elementary edit operations.

In order to compute $d(x, y)$ for two strings x and y of length m and n respectively, we use a two-dimensional table t of $m + 1$ rows and $n + 1$ columns such that $t[i, j] = d(x[0\,..\,i], y[0\,..\,j])$ for $i = 0, \ldots, m - 1$ and $j = 0, \ldots, n - 1$. It follows $d(x, y) = t[m - 1, n - 1]$.

The values of the table t can be computed by the following recurrence formulas:

$$t[i, j] = \min \begin{cases} t[i - 1, j - 1] + Sub(x[i], y[j]) \\ t[i - 1, j] + Del(x[i]) \\ t[i, j - 1] + Ins(y[j]) \end{cases}$$

for $i = 0, 1, \ldots, m - 1$ and $j = 0, 1, \ldots, n - 1$. And there are margin initializations:

$$t[-1, -1] = 0,$$

$$t[i, -1] = t[i - 1, -1] + Del(x[i]),$$

$$t[-1, j] = t[-1, j - 1] + Ins(y[j])$$

for $i = 0, 1, \ldots, m - 1$ and $j = 0, 1, \ldots, n - 1$. The value at position (i, j) in the table t only depends on the values at the three neighbor positions $(i - 1, j - 1)$, $(i - 1, j)$ and $(i, j - 1)$ (see [12]).

The direct application of the above recurrence formula gives an exponential time algorithm to compute $t[m - 1, n - 1]$. However the whole table t can be computed in quadratic time, technique known as "dynamic programming". This is a general technique that is used to solve the different kinds of alignments.

An optimal alignment (with minimal cost) can then be produced. It consists in tracing back the computation of the values of the table t from position $(m - 1, n - 1)$ to position $(-1, -1)$.

2.2 Local alignment

A local alignment of two strings x and y consists in finding the segment of x that is closer to a segment of y. The notion of distance used to compute global alignments cannot be used in that case since the segments of x closer to segments of y would only be the empty segment or individual symbols. This is why a notion of similarity is used based on a scoring scheme for edit operations.

A score (instead of a cost) is associated to each elementary edit operation. For $a, b \in \Sigma$: $Sub_S(a, b)$ denotes the score of substituting the symbol b for the symbol a (the more similar the two symbols a and b the higher the substitution score and the smallest the cost which cannot be smaller than 0 for two equal symbols), $Del_S(a)$ denotes the score of deleting the symbol a, $Ins_S(a)$ denotes the score of inserting the symbol a. For two symbols a and b, a positive value of $Sub_S(a, b)$ means that the two symbols are close to each other, and a negative value of $Sub_S(a, b)$ means that the two symbols are far apart.

We can now define the edit score of two strings x and y by $s(x, y) =$ maximal similarity between a segment of x and a segment of y.

In order to compute $s(x, y)$ for two strings x and y of length m and n respectively, we make use of a two-dimensional table t_S of $m+1$ rows and $n+1$ columns such that $t_S[i, j] = \max\{s(x[\ell\,..\,i], y[k\,..\,j]) \mid 0 \leq \ell \leq i \text{ and } 0 \leq k \leq j\} \cup \{0\}$, for $i = 0, \ldots, m-1$ and $j = 0, \ldots, n-1$. Therefore, $s(x, y) =$ maximal value in t_S.

The values of the table t_S can be computed by the following recurrence formula:

$$t_S[-1, -1] = t_S[i, -1] = t_S[-1, j] = 0$$

and

$$t_S[i, j] = \max \begin{cases} t_S[i-1, j-1] + Sub_S(x[i], y[j])\,, \\ t_S[i-1, j] + Del_S(x[i])\,, \\ t_S[i, j-1] + Ins_S(y[j])\,, \\ 0\,. \end{cases}$$

for $i = 0, 1, \ldots, m-1$ and $j = 0, 1, \ldots, n-1$.

Computing the values of t_S for a local alignment of x and y can be done in $O(mn)$ time and space complexity. Recovering a local alignment can be done in a way similar to what is done in the case of a global alignment, but the traceback procedure must start at a position of a maximal value in t_S rather than at position $(m-1, n-1)$.

3 Aligning 2D patterns

Let us now assume that we consider two rectangular patterns $X = X[0\,..\,m_1 - 1, 0\,..\,n_1 - 1]$ and $Y = Y[0\,..\,m_2 - 1, 0\,..\,n_2 - 1]$, of size $M = m_1 \times n_1$ and $N = m_2 \times n_2$ respectively. Each element $X[i, j]$ with $0 \leq i \leq m_1 - 1$ and $0 \leq j \leq n_1 - 1$ and $Y[k, \ell]$ with $0 \leq k \leq m_2 - 1$ and $0 \leq \ell \leq n_2 - 1$ belongs to the alphabet Σ.

We want to compute the minimum cost of operations of insertion, deletion or substitution of individual symbols to transform X into Y. Symbols can be inserted, deleted and substituted separately or in sub-rows or sub-columns.

3.1 Global alignment

Aligning X and Y using dynamic programming consists in generalizing the recurrence formulas used for string alignment.

To that aim we need four two-dimensional tables D_R, D_C, I_R and I_C defined as follows:

$$D_R[i,j] = \sum_{p=0}^{j} Del(X[i,p]) \qquad D_C[i,j] = \sum_{p=0}^{i} Del(X[p,j])$$

$$I_R[i,j] = \sum_{p=0}^{j} Ins(Y[i,p]) \qquad I_C[i,j] = \sum_{p=0}^{i} Ins(Y[p,j])$$

for $0 \leq i \leq m_1 - 1$, $0 \leq j \leq n_1 - 1$, $0 \leq i \leq m_2 - 1$ and $0 \leq j \leq n_2 - 1$.

In words, $D_R[i,j]$ is the cost of the deletion of the prefix of length $j+1$ of row i of X, $D_C[i,j]$ is the cost of the deletion of the prefix of length $i+1$ of column j of X, $I_R[i,j]$ is the cost of the insertion of the prefix of length $j+1$ of row i of Y and $I_C[i,j]$ is the cost of the insertion of the prefix of length $i+1$ of row j of Y. The tables D_R and D_C can be computed in time and space $O(m_1 \times n_1)$. The tables I_R and I_C can be computed in time and space $O(m_2 \times n_2)$.

We will also use two four-dimensional tables R and C of size $m_1 \times n_1 \times m_2 \times n_2$ defined as follows: $R[i,j,k,\ell] = d(X[i,0\,..\,j], Y[k,0\,..\,\ell])$, and $C[i,j,k,\ell] = d(X[0\,..\,i,j], Y[0\,..\,k,\ell])$. In words, $R[i,j,k,\ell]$ contains the distance between the prefix of length $j+1$ of row i of X and the prefix of length $\ell+1$ of row k of Y. Similarly, $C[i,j,k,\ell]$ contains the distance between the prefix of length $i+1$ of column j of X and the prefix of length $k+1$ of column ℓ of Y. The two tables R and C can be computed in time and space $O(m_1 \times n_1 \times m_2 \times n_2)$.

Then, we use a four dimensional table T of size $(m_1+1) \times (n_1+1) \times (m_2+1) \times (n_2+1)$ defined as follows: $T[i,j,k,\ell] = \min\{\text{cost of } \gamma \mid \gamma \in \Gamma_{X,Y}\}$ where $\Gamma_{X,Y}$ is the set of all the sequences of edit operations that transform X into Y, and the cost of an element $\gamma \in \Gamma_{x,y}$ is the sum of the costs of its elementary edit operations.

The values of the table T can be computed as follows for $0 \leq i \leq m_1 - 1$, $0 \leq j \leq n_1 - 1$, $0 \leq k \leq m_2 - 1$ and $0 \leq \ell \leq n_2 - 1$ (see Figure 1):

$$T[i,j,k,\ell] = \min \begin{cases} T[i-1,j,k,\ell] + D_R[X[i,0\,..\,j]] \\ T[i,j-1,k,\ell] + D_C[X[0\,..\,i,j]] \\ T[i,j,k-1,\ell] + I_R[Y[k,0\,..\,\ell]] \\ T[i,j,k,\ell-1] + I_C[Y[0\,..\,k,\ell]] \\ T[i-1,j,k-1,\ell] + R[i,j,k,\ell] \\ T[i,j-1,k,\ell-1] + C[i,j,k,\ell] \\ T[i-1,j-1,k-1,\ell-1] + \\ \qquad C[i,j-1,k,\ell-1] + R[i,j,k,\ell] \\ T[i-1,j-1,k-1,\ell-1] + \\ \qquad C[i,j,k,\ell] + R[i-1,j,k-1,\ell] \end{cases} \tag{1}$$

We have the following margin initializations: $T[-1,j,k,\ell] = T[i,-1,k,\ell] = (k+1)\times(\ell+1)$ and $T[i,j,-1,\ell] = T[i,j,k,-1] = (i+1)\times(j+1)$ for $0 \leq i \leq m_1 - 1$, $0 \leq j \leq n_1 - 1$, $0 \leq k \leq m_2 - 1$ and $0 \leq \ell \leq n_2 - 1$.

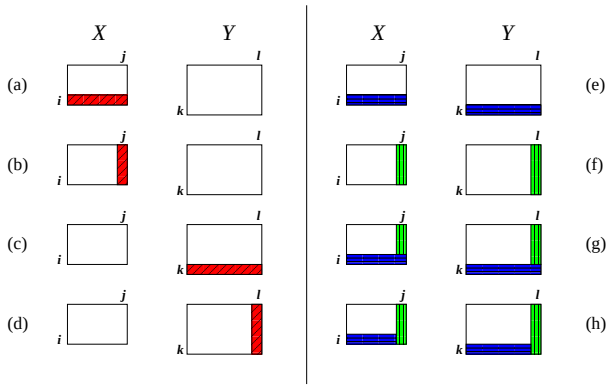

Figure 1 When processing $X[i,j]$ and $Y[k,\ell]$: the eight different ways to align $X[0\,..\,i,0\,..\,j]$ with $Y[0\,..\,k,0\,..\,\ell]$. White parts are similar; red parts are deleted or inserted; blue and green parts are substituted. (a): $X[0\,..\,i-1,0\,..\,j]$ is similar to $Y[0\,..\,k,0\,..\,\ell]$: deletion of $X[i,0\,..\,j]$; (b): $X[0\,..\,i,0\,..\,j-1]$ is similar to $Y[0\,..\,k,0\,..\,\ell]$: deletion of $X[0\,..\,i,j]$; (c): $X[0\,..\,i,0\,..\,j]$ is similar to $Y[0\,..\,k-1,0\,..\,\ell]$: insertion of $Y[k,0\,..\,\ell]$; (d): $X[0\,..\,i,0\,..\,j]$ is similar to $Y[0\,..\,k,0\,..\,\ell-1]$: insertion of $Y[0\,..\,k,\ell]$; (e): $X[0\,..\,i-1,0\,..\,j]$ is similar to $Y[0\,..\,k-1,0\,..\,\ell]$: substitution of $X[i,0\,..\,j]$ by $Y[k,0\,..\,\ell]$; (f): $X[0\,..\,i,0\,..\,j-1]$ is similar to $Y[0\,..\,k,0\,..\,\ell-1]$: substitution of $X[0\,..\,i,j]$ by $Y[0\,..\,k,\ell]$; (g): $X[0\,..\,i-1,0\,..\,j-1]$ is similar to $Y[0\,..\,k-1,0\,..\,\ell-1]$: substitution of $X[i,0\,..\,j]$ by $Y[k,0\,..\,\ell]$ and substitution of $X[0\,..\,i-1,j]$ by $Y[0\,..\,k-1,\ell]$; (h): $X[0\,..\,i-1,0\,..\,j-1]$ is similar to $Y[0\,..\,k-1,0\,..\,\ell-1]$: substitution of $X[i,0\,..\,j-1]$ by $Y[k,0\,..\,\ell-1]$ and substitution of $X[0\,..\,i,j]$ by $Y[0\,..\,k,\ell]$.

Then a traceback can be performed from $T[m_1-1,n_1-1,m_2-1,n_2-1]$ as in the two sequence comparison case.

The global alignment of two rectangular patterns of respective size $M = m_1 \times n_1$ and $N = m_2 \times n_2$ can be performed in time and space $O(M \times N)$. Example:

With the following rectangular patterns

X	0	1	2
0	**A**	**B**	**C**
1	**D**	**E**	**F**
2	**G**	**H**	**I**
3	**J**	**K**	**L**

and

Y	0	1
0	**E**	**C**
1	**H**	**I**
2	**K**	**L**

considering unit cost edit operations ($Ins(a) = Del(a) = 1$, $Sub(a,a) = 0$ and $Sub(a,b) = 1$ for $a,b \in \Sigma$ such that $a \neq b$), one possible script when tracing back from $T[3,2,2,1] = 6$ is:

Operation	Cost
Substitution of $X[0\,..\,3,2]$ by $Y[0\,..\,2,1]$	1
Substitution of $X[0\,..\,3,1]$ by $Y[0\,..\,2,0]$	1
Deletion of $X[0\,..\,3,0]$	4

3.2 Local alignment

A local alignment of two two-dimensional matrices X and Y consists in finding a portion X' of X and a portion Y' of Y such that the similarity between X' and Y' is maximal (among all the pairs of portions of X and Y). The set of operations that we consider here is: the deletion of a single symbol of X, the insertion of a single symbol of Y, the deletion of a sub-row of X, the insertion of a sub-row of Y, the deletion of a sub-column of X, the insertion of a sub-column of Y, the substitution of a single symbol of X by a single symbol of Y, the substitution of a sub-row of X by a sub-row of Y and the substitution of a sub-column of X by a sub-column of Y.

To compute a local alignment between two rectangular two-dimensional matrices we will use two four-dimensional tables R_S and C_S of size $m_1 \times n_1 \times m_2 \times n_2$ defined as follows: $R_S[i,j,k,\ell] = s(X[i,0\,..\,j], Y[k,0\,..\,\ell])$, and $C_S[i,j,k,\ell] = s(X[0\,..\,i,j], Y[0\,..\,k,\ell])$. Those two tables can be computed using the usual recurrence formulas for strings in time and space $O(m_1 \times n_1 \times m_2 \times n_2)$.

Then we use a four-dimensional table T_S of size $(m_1+1) \times (n_1+1) \times (m_2+1) \times (n_2+1)$.

Let us denote: $r = R_S[i,j,k,\ell]$, $c = C_S[i,j,k,\ell]$, $r' = R_S[i-1,j,k-1,\ell]$, $c' = C_S[i,j-1,k,\ell-1]$ and $q = Del_S(X[i,j]) + Ins_S(Y[k,\ell])$.

The values of the table T_S can be computed by the equation given below (see Figure 2):

$$T_S[i,j,k,\ell] = \max \begin{cases} T_S[i-1,j,k,\ell] + Del_S(X[i,j]) \\ T_S[i,j-1,k,\ell] + Del_S(X[i,j]) \\ T_S[i,j,k-1,\ell] + Ins_S(Y[k,\ell]) \\ T_S[i,j,k,\ell-1] + Ins_S(Y[k,\ell]) \\ T_S[i-1,j,k-1,\ell] + (r \text{ if } r \neq 0 \text{ otherwise } q) \\ T_S[i,j-1,k,\ell-1] + (c \text{ if } c \neq 0 \text{ otherwise } q) \\ T_S[i-1,j-1,k-1,\ell-1] + \\ \qquad (c'+r \text{ if } c', r \neq 0 \text{ otherwise } q) \\ T_S[i-1,j-1,k-1,\ell-1] + \\ \qquad (c+r' \text{ if } c, r' \neq 0 \text{ otherwise } q) \\ 0 \end{cases} \tag{2}$$

for $0 \leq i \leq m_1 - 1$, $0 \leq j \leq n_1 - 1$, $0 \leq k \leq m_2 - 1$ and $0 \leq \ell \leq n_2 - 1$.

We have the following margin initializations: $T_S[-1,j,k,\ell] = T_S[i,-1,k,\ell] = 0$ and $T_S[i,j,-1,\ell] = T_S[i,j,k,-1] = 0$ for $0 \leq i \leq m_1 - 1$, $0 \leq j \leq n_1 - 1$, $0 \leq k \leq m_2 - 1$ and $0 \leq \ell \leq n_2 - 1$.

The values of the table T_S of size $(m_1+1) \times (n_1+1) \times (m_2+1) \times (n_2+1)$ for detecting similar rectangular sub-patterns can be performed in time and space $O(m_1 \times n_1 \times m_2 \times n_2)$.

The traceback procedure must start at a position of a maximal value in T_S rather than at position $(m_1-1, n_1-1, m_2-1, n_2-1)$.

Example: With the following score system: $Sub_S(a,a) = 1$, $Sub_S(a,b) = -1$ and $Ins_S(a) = Del_S(a) = -1$, for $a, b \in \Sigma$ such that $a \neq b$), Figure 3 shows on the left

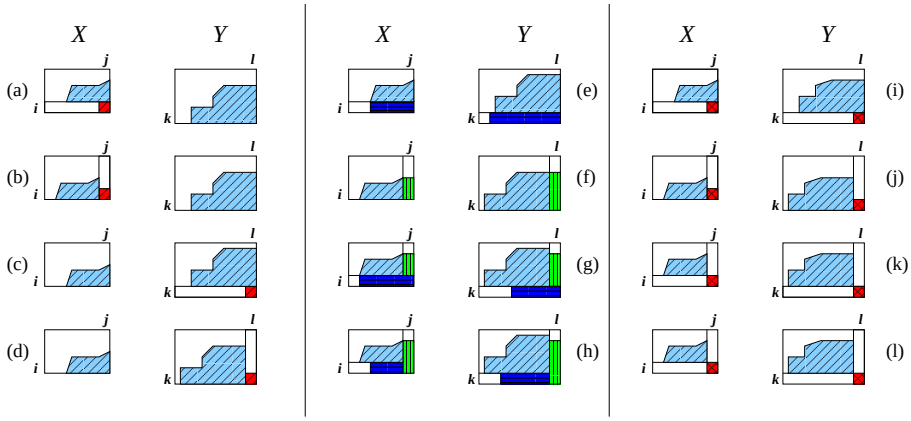

Figure 2 When processing $X[i,j]$ and $Y[k,\ell]$: the different ways to locally align $X[0\,..\,i,0\,..\,j]$ with $Y[0\,..\,k,0\,..\,\ell]$. Light blue parts are similar; red parts are deleted or inserted; dark blue and green parts are substituted. (a): deletion of $X[i,j]$ and alignment of $x[0\,..\,i-1,0\,..\,j]$ with $Y[0\,..\,k,0\,..\,\ell]$; (b): deletion of $X[i,j]$ and alignment of $x[0\,..\,i,0\,..\,j-1]$ with $Y[0\,..\,k,0\,..\,\ell]$; (c): insertion of $Y[k,\ell]$ and alignment of $x[0\,..\,i,0\,..\,j]$ with $Y[0\,..\,k-1,0\,..\,\ell]$; (d): insertion of $Y[0,\ell]$ and alignment of $x[0\,..\,i,0\,..\,j]$ with $Y[0\,..\,k,0\,..\,\ell-1]$; (e): substitution of $X[i,j'\,..\,j]$ by $Y[k,\ell'\,..\,\ell]$ and alignment of $x[0\,..\,i-1,0\,..\,j]$ with $Y[0\,..\,k-1,0\,..\,\ell]$; (f): substitution of $X[i'\,..\,i,j]$ by $Y[k'\,..\,k,\ell]$ and alignment of $x[0\,..\,i,0\,..\,j-1]$ with $Y[0\,..\,k,0\,..\,\ell-1]$; (g): substitution of $X[i,j'\,..\,j]$ by $Y[k,\ell'\,..\,\ell]$ and substitution of $X[i'\,..\,i-1,j]$ by $Y[k'\,..\,k-1,\ell]$ and alignment of $x[0\,..\,i-1,0\,..\,j-1]$ with $Y[0\,..\,k-1,0\,..\,\ell-1]$; (h): substitution of $X[i,j'\,..\,j-1]$ by $Y[k,\ell'\,..\,\ell-1]$ and substitution of $X[i'\,..\,i,j]$ by $Y[k'\,..\,k,\ell]$ and alignment of $x[0\,..\,i-1,0\,..\,j-1]$ with $Y[0\,..\,k-1,0\,..\,\ell-1]$. (i): substitution of $X[i,j]$ by $Y[k,\ell]$ and alignment of $x[0\,..\,i-1,0\,..\,j]$ with $Y[0\,..\,k-1,0\,..\,\ell]$; (j): substitution of $X[i,j]$ by $Y[k,\ell]$ and alignment of $x[0\,..\,i,0\,..\,j-1]$ with $Y[0\,..\,k,0\,..\,\ell-1]$; (k): substitution of $X[i,j]$ by $Y[k,\ell]$ and alignment of $x[0\,..\,i-1,0\,..\,j-1]$ with $Y[0\,..\,k-1,0\,..\,\ell-1]$; (l): substitution of $X[i,j]$ by $Y[k,\ell]$ and alignment of $x[0\,..\,i-1,0\,..\,j-1]$ with $Y[0\,..\,k-1,0\,..\,\ell-1]$; with $0\leq i'\leq i$, $0\leq j'\leq j$, $0\leq k'\leq k$ and $0\leq \ell'\leq \ell$.

the two patterns to align and on the right the best local alignment where symbol **c** on the third line of the second pattern is inserted, while symbols **ghi** of the third line of the first pattern are substituted by symbols **aba** of the fourth line of the second pattern.

This method can be applied in the four orientations and the four best local alignments can then be merged. All this can still be performed with the same complexities.

4 Application

The algorithm was tested on dialogues between parents and children during the narration of child stories. Each utterance has been transcribed and coded according to a grid [10, 11]. For each dialogue, we therefore obtain a coding matrix whose number of columns is fixed whereas the number of lines depends on the length of the dialogue. Then, we applied the two dimensional pattern alignment

```
A b C J e f g T i j 5 l m n o
B b C d E f g h i j k l m n o
B b C J L f g h i j k l m n o
A b C d E f g h i j k l m n o

A b C J e f g T i j 5 l m n
B b C d E f g h i , ; : ! ?
x y z t u v w a c e p q r s
B b C J L f a b a b a c a b
A b C d E f g h i j k l m n
a j g j h a j h i j a j j j

A b C J e f g T i
B b C d E f g h i
              c
B b C J L f a b a
A b C d E f g h i j k l m n
```

Figure 3 Local alignment of 2D patterns.

algorithm over all the pairs of different dialogue matrices, in order to extract regularities in the verbal behaviours of parents and children.

4.1 Coding of the dialogues

The grid used to code the dialogues was designed as part of a psychological study on interpretations about the behaviour of a character who is confronted with a false belief [10, 11]. The aim of this study is to identify dialogical, semantical and pragmatical characteristics of adult's discourses according to the age of the child they speak to [21].

25	P	don't worry	A	P	E	)	]
26	P	so they hide	A	P	B	)	]
27	P	they are looking for	a	[	f	)	]
28	P	who could have taken away the crown	q	[	f	)	]
29	C	it's in ! The crown is inside the box !	a	[	f	)	]
30	P	so they are suspecting a lot of people, Cornelius, Celeste, the old lady	A	P	Y	C	J
31	P	who could have stolen the crown	q	[	f	)	]
32	C	the crown, it's in!	a	[	f	)	]
33	P	do you believe it?	Q	H	K	)	]
34	C	yes	a	[	f	)	]
35	P	but Babar doesn't know that it's in	A	P	N	O	J
36	P	so he says that the crown is a bomb	A	P	N	C	J
37	P	or I don't know what else	A	R	N	)	]
38	C	the crown	a	[	f	)	]

Figure 4 Example of coded dialogue

This grid is composed of five columns, each column includes between 2 and 7 encoding terms (6, 3, 7, 2, 2) (see Figure 4):

- The terms of the first column are related to the nature of the utterance. It can be either an affirmation (a or A), a query (q or Q), a request to pay attention to the story (D), or a demand for general attention (G). A capital letter (A, Q, D and G) corresponds to the presence of a mental state.

- The second column expresses to what the utterance is referring. The utterance can refer to the character (P), the interlocutor (H) or the speaker (R).

- The third column is dedicated to mental states. The interlocutors can express an emotion (E), a volition (V), an observable or a non-observable cognition (B or N), an epistemic statement (K), an assumption (Y) or a surprise (S). The surprise is distinguished from other emotions because of its link with the incidental belief.

- The last two columns represent explanations using cause / consequence (C), opposition (O) or empathy (M), which can be applied either to explain the story (J), or to precise a situation with a personal context (F).

For each of the five columns, a symbol is used to represent the absence of information: (, [, {,),] respectively.

For example, line 35 of dialogue of Figure 4: "*but Babar doesn't know that it's in*" is coded in the following way: this statement contains a mental state, thus A appears in the first column; this mental state refers to a character ("*Babar*"), thus P appears in the second column; the corresponding mental state ("*know*") refers to non observable cognition, thus N appears in the third column; "*but*" denotes a justification by opposition, thus O appears in the fourth column; finally, the context refers to the story itself, thus J appears in the fifth column.

4.2 Local alignment of 2D patterns using dynamic programming

The local two dimensional pattern alignment algorithm was applied over all the pairs of dialogue matrices in order to identify approximate (or fuzzy) patterns. Those pairs can be either pairs of different dialogues (inter-dialogue alignment) or twice the same dialogue (intra-dialogue alignment).

A vector of insertion scores, a vector of deletion scores as well as a substitution matrix were specifically designed for this application, in order to express the elementary operations of edition (see Figure 5 for the substitution matrix; if $a \in \{(, [, \{,),]\}$ then $Sub_S(a, b) = Sub_S(b, a) = -4$ for $a \neq b$ and $Sub_S(a, a) = -2$). The substitution matrix expresses how each coding element is linked to another. A low value between two coding elements indicates that they cannot be substituted one by the other. On the contrary, a high value between two coding elements means that these elements are similar and can be substituted one by the other.

We obtain a series of pairs of alignments and the corresponding similarity score for each pair. This technique enabled us to identify approximate/fuzzy patterns repeated inside a unique dialogue or within two different dialogues. An example of extracted patterns along the alignment process is given Figure 6 and the corresponding dialogues are given Figure 7. The pattern alignment appears in light gray. Each code is written in blue bold when corresponding to a mental state or a reference (second column or third column) and is written in red italic when corresponding to explanation and context (fourth column or fifth column). The same colors and shapes are used in the corresponding dialogues.

The evaluation of the scientific interest of the discovered patterns is, of course, domain dependent. As a reminder, the proposed application is linked with a

	a	q	A	Q	G	D	P	R	H	C	O	M	J	F	E	B	N	V	K	Y	S
a	10																				
q	9	10																			
A	10	9	10																		
Q	9	10	9	10																	
G	2	8	2	8	10																
D	2	8	2	8	10	10															
P	0	0	0	0	0	0	10														
R	0	0	0	0	0	0	9	10													
H	0	0	0	0	0	0	7	8	10												
C	0	0	0	0	0	0	2	2	2	10											
O	0	0	0	0	0	0	2	2	2	9	10										
M	0	0	0	0	0	0	2	8	2	8	8	10									
J	0	0	0	0	0	0	7	4	3	1	1	2	10								
F	0	0	0	0	0	0	4	7	3	1	1	7	9	10							
E	0	0	0	0	0	0	5	7	5	3	3	8	2	4	10						
B	0	0	0	0	0	0	5	5	5	3	3	3	4	4	8	10					
N	0	0	0	0	0	0	5	5	5	3	3	3	4	4	8	10	10				
V	0	0	0	0	0	0	5	5	5	3	3	3	2	4	9	7	7	10			
K	0	0	0	0	0	0	7	5	5	3	6	3	4	4	8	7	8	7	10		
Y	0	0	0	0	0	0	5	5	5	4	3	3	4	4	8	7	8	7	9	10	
S	0	0	0	0	0	0	5	5	5	3	3	6	3	4	9	7	7	7	9	8	10

Figure 5 Partial substitution matrix

Dialogue b7

25	A	P	E	)	]
26	A	**P**	**B**	)	]
27	a	[	{	)	]
28	q	[	{	)	]
29	a	[	{	)	]
30	A	**P**	**Y**	C	J
31	q	[	{	)	]
32	a	[	{	)	]
33	Q	**H**	**K**	)	]
34	a	[	{	)	]
35	A	**P**	**N**	*O*	*J*
36	A	**P**	**N**	*C*	*J*
37	A	**R**	**N**	)	]
38	a	[	f	)	]

Dialogue b9

8	q	[	{	)	]
9	A	**P**	**B**	)	]
10	q	[	{	)	]
11	q	[	{	)	]
12	q	[	{	)	]
13	A	**P**	**B**	)	]
14	a	[	{	)	]
15	A	**P**	**N**	)	]
16	q	[	{	)	]
17	a	[	{	)	]
18	a	[	{	)	]
19	A	**P**	**V**	)	]
20	A	**P**	**B**	*O*	*J*
21	A	**P**	**S**	*O*	*J*
22	a	[	{	)	]
23	a	[	{	)	]

Figure 6 Example of local inter-dialogue pattern alignment

psychological study which aims at identifying characteristics of adults' discourse when telling a story to a child. In this example, inside a series of lines, a L-shaped pattern appears:

- The vertical part of the L is composed of a set of elements that code a mental state associated with a reference.
- The horizontal part of the L contains a justification associated with a mental state.

The interest of this pattern, in a psychological point of view, is that it seems to characterize both dialogic organization and content. In this example, it appears that parents ensure to prepare a justification of a situation using mental states.

Dialogue b7

25	P	don't worry
26	P	so **they hide**
27	P	they are looking for
28	P	who could have taken away the crown
29	C	it's in ! The crown is inside the box !
30	P	so **they are suspecting** a lot of people, Cornelius, Celeste, the old lady
31	P	who could have stolen the crown
32	C	the crown, it's in!
33	P	**do you believe it?**
34	C	yes
35	P	*but* Babar **doesn't know** that it's in
36	P	*so* **he says** that the crown is a bomb
37	P	or **I don't know** what else
38	C	the crown

Dialogue b9

9	P	but where is it?
10	P	Babar **sees** that Cornelius has given a pack to a stranger
11	P	what's inside?
12	P	but who is this masked stranger?
13	P	who has stolen the crown?
14	P	Babar **uncovers** her!
15	P	she's queen Celeste!
16	P	**he asks** many questions
17	P	so why did the Queen disguised?
18	P	so Babar goes to see the old lady to ask her
19	C	yes
20	P	The old lady **doesn't want** him inside!
21	P	*but* just behind her, she **hides**
22	P	a **surprise** for him *in fact*
23	P	and then Babar gets back home

Figure 7 Inter-dialogue patterns

Indeed, during a first step, they describe and explain the mental states involved. Then, in a second step, they clarify the cause, consequence or opposition that explain the mental states of a character and the character's behaviour.

Therefore, the algorithm we design enables us to discover a set of alignments among the corpus of dialogues, such as the example presented above. These alignments, still under expertise evaluation, characterize the behaviour of parents and children during the proposed narration task.

4.3 Software implementation

A software implementation in C++ of the previously described algorithm can be downloaded on sourceforge (`http://sourceforge.net/projects/apsad/`). It generates a list of pairs of alignments ordered according to their similarity scores and saved into an XML file. The designed HMI enables to visualize the saved alignments inside any kind of web browser.

5 Conclusion and future work

We describe a new heuristic to discover patterns in annotated dialogues using dynamic programming. The design method is sufficiently generic to align any kind of two dimensional patterns in quadratic time and space. This method enables the computation of global and local alignments. The method has been applied to dialogues between parents and children during the narration of child stories.

Many questions arise, particularly concerning the algorithm part. Is it possible to align 2D patterns in sub-quadratic time as for sequences? (see [13]). Is it possible to develop a heuristic similar to BLAST ([1]) to quickly align a 2D pattern against a data bank of 2D patterns?

As an expertise evaluation task is necessary to evaluate the scientific interest of any discovered pattern, another challenge consists in designing a tool to visualize

the two dimensional computed alignments. It is also necessary to facilitate the expert evaluation by proposing a statistical evaluation of the relevance of the results.

This raise a particular difficulty: a reference pattern has to be chosen among each alignment which can be extracted from two different dialogues or extracted from two different parts of the same dialogue. Moreover, several different alignments can also cover the same part of a dialogue, thus linking a set of different patterns. A reference pattern has also to be chosen for the complete set. Therefore, all the reference patterns, selected if of interest, will be referenced in a database and searched for in all the dialogues, using a two dimensional approximate pattern matching such as presented in [5].

On a longer term basis, a classification process has to be applied on the corpus of dialogues in order to separate sets of dialogues according to the reference patterns they contain. As an example of application, we aim at classifying the dialogues, and therefore the parents' behaviours, according to the age of the child they speak to.

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[2] A. Amir and M. Farach. Two dimensional matching. In A. Apostolico and Z. Galil, editors, *Pattern Matching Algorithms*, chapter 9, pages 267–292. Oxford University Press, 1997.

[3] A. Amir, T. Hartman, O. Kapah, R. Shalom, and D. Tsur. Generalized LCS. In N. Ziviani and R. Baeza-Yates, editors, *Proc of. SPIRE'07*, volume 4726 of *Lecture Notes in Computer Science*, pages 50–61, Santiago, Chile, 2007. Springer.

[4] A. N. Arslan. A largest common d-dimensional subsequence of two d-dimensional strings. In E. Csuhaj-Varjú and Z. Ésik, editors, *Proc. of FCT'07*, volume 4639 of *Lecture Notes in Computer Science*, pages 40–51, Budapest, Hungary, 2007. Springer.

[5] R. Baeza-Yates and G. Navarro. Fast two-dimensional approximate pattern matching. In C. Lucchesi and A. Moura, editors, *Proc. of LATIN'98: Theoretical Informatics*, volume 1380 of *Lecture Notes in Computer Science*, pages 341–351, Campinas, Brazil, 1998. Springer.

[6] R. Baeza-Yates and M. Régnier. Fast two-dimensional pattern matching. *Inf. Process. Lett.*, 45(1):51–57, 1993.

[7] R. A. Baeza-Yates. Similarity in two-dimensional strings. In W.-L. Hsu and M.-Y. Kao, editors, *Proc. of COCOON'98*, volume 1449 of *Lecture Notes in Computer Science*, pages 319–328, Taipei, Taiwan, R.o.C., 1998. Springer.

[8] R. S. Bird. Two-dimensional pattern matching. *Inf. Process. Lett.*, 6(5):168–170, 1977.

[9] H. Bunt. The DIT++ taxonomy for functional dialogue markup. In D. Heylen, C. Pelachaud, R. Catizone, and D. Traum, editors, *AAMAS 2009 Workshop, Towards a Standard Markup Language for Embodied Dialogue Acts*, pages 13–24, Budapest, Hungary, 2009.

[10] É. Chanoni. *Rôle du langage dans le développement de la théorie de lesprit chez les enfants de 3 à 5 ans: contexte verbal et contexte narratif.* PhD thesis, Presse Universitaire de Lille, 2004.

[11] É. Chanoni. Comment les mères racontent une histoire de fausses croyances à leur enfant de 3 à 5 ans ? *Enfance*, 63(2), 2009.

[12] M. Crochemore, C. Hancart, and T. Lecroq. *Algorithms on strings.* Cambridge University Press, 2007.

[13] M. Crochemore, G. M. Landau, and M. Ziv-Ukelson. A subquadratic sequence alignment algorithm for unrestricted scoring matrices. *SIAM J. Comput*, 32(6):1654–1673, 2003.

[14] K. Fredriksson, G. Navarro, and E. Ukkonen. Optimal exact and fast approximate two dimensional pattern matching allowing rotations. In A. Apostolico and M. Takeda, editors, *Proc. of CPM'02*, volume 2373 of *Lecture Notes in Computer Science*, pages 235–248, Fukuoka, Japan, 2002. Springer.

[15] K. Krithivasan and R. Sitalakshmi. Efficient two-dimensional pattern matching in the presence of errors. *Inf. Sci.*, 43(3):169–184, 1987.

[16] H. Mazouzi, A. El-Fallah-Seghrouchni, and S. Haddad. Open protocol design for complex interactions in multi-agent systems. In *Proc. of AAMAS'02*, pages 517–526, Bologna, Italy, 2002. ACM.

[17] J. C. Na, R. Giancarlo, and K. Park. On-line construction of two-dimensional suffix trees in $O(n^2 \log n)$ time. *Algorithmica*, 48(2):173–186, 2007.

[18] A. Pauchet, A. El-Fallah-Seghrouchni, and N. Chaignaud. Simulating a human cooperative problem solving. In H.-D. Burkhard, G. Lindemann, R. Verbrugge, and L. Varga, editors, *Proc. of CEEMAS'07*, volume 4696 of *Lecture Notes in Computer Science*, Leipzig, Germany, 2007. Springer.

[19] D. Sankoff and J. B. Kruskal. *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison.* Cambridge University Press, 1999. 2nd edition.

[20] FIPA. FIPA request interaction protocol specification. Technical report, Foundation for Intelligent Physical Agents, 2002. Available at `http://www.fipa.org`.

[21] H. Wimmer and J. Perner. Beliefs about beliefs: representation and constraining function of the wrong beliefs in young childrens understandig of deception. *Cognition*, 13(1):103–128, 1983.

[22] T. Winograd and F. Flores. *Understanding Computers and Cognition: A New Foundation for Design.* Ablex, 1986.