Extraction and Clustering of Two-Dimensional Dialogue Patterns

Zacharie Alès, Alexandre Pauchet, Arnaud Knippel

Extraction and clustering of two-dimensional dialogue patterns

Zacharie Ales
Unité de Mathématiques Appliquées, ENSTA ParisTech, Université Paris-Saclay, 91120 Palaiseau, France
Université d'Avignon et des Pays de Vaucluse, LIA, EA4128
zacharie.ales@ensta-paristech.fr

Alexandre Pauchet
Normandie Univ, INSA Rouen, UNIHAVRE, UNIROUEN, LITIS, 76000 Rouen, France
alexandre.pauchet@insa-rouen.fr

Arnaud Knippel
Normandie Univ, INSA Rouen, LMI, 76000 Rouen, France
arnaud.knippel@insa-rouen.fr

Abstract

This article proposes a two-step methodology to ease the identification of dialogue patterns in a corpus of annotated dialogues. The annotations of a given dialogue are represented within a two-dimensional array whose lines correspond to the utterances of the dialogue ordered chronologically.

The first step of our methodology consists in extracting recurrent patterns. To that end, we adapt a dynamic programming algorithm used to align two-dimensional arrays by reducing its complexity and improving its trace-back procedure. During the second step, the obtained patterns are clustered using various heuristics from the literature.

As evaluation process, our method is applied onto a corpus of annotated dialogues between a parent and her child in a storytelling context. The obtained partitions of dialogue patterns are evaluated by an expert in child development of language to assess how the methodology helps the expert into explaining the child behaviors.

The influence of the method parameters (clustering heuristics, minimum extraction score, number of clusters and substitution score array) are studied. Dialogue patterns that manual extractions have failed to detect are highlighted by the method and the most efficient values of the parameters are therefore determined.

Keywords: Regularity extraction; Pattern extraction; Clustering; Dialogue modeling.

1 Introduction

More than forty years after the creation of the first famous chatter-bot ELIZA[44], dialogue systems able to openly interact with humans still need to be designed. Currently, when interacting with conversational agents, humans must adapt to rigid schemes and linear dialogue management[24, 42]. For instance, interactive voice response systems restrict human interaction to keywords at specific times. In other words, the dialogue is completely directed by the system.

Humans, creatures of habits and conventions, are used to regular activities and acquired social behaviors. Dialogue which is a typically human activity is sprinkled with regularities. We share the point of view of Orkin *et al.*[30] and Dubuisson Duplessis *et al.*[13] that recurrent dialogue patterns occurring in Human-Human interaction can be exploited to model Human-Machine interaction. Indeed, identification of patterns may first help to analyse the containing dialogue and, secondly, such patterns also constitute a basis to define dialogue games[20, 26, 27]. Consequently, the extraction of regularities from corpora of dialogues can significantly help the creation of such dialogue management systems[1].

Existing approaches to extract recurrent dialogue patterns are strongly focused on the goal of the dialogue and thus likely to underestimate valuable additional information. This is especially true in pragmatics in which all the dialogue moves are considered at the task level. In many studies, only one type of annotations, dedicated to the task, is associated to each dialogue utterance[12, 13, 30]. However, as outlined by Bunt[7], dialogue management involves multilevel aspects. As a consequence, in his annotation scheme Dit++[6], only one dimension among the ten is dedicated to the task. Moreover, some messages can be sent and understood through various modalities. An agreement can, for example, be expressed by saying "yes" or by nodding the head. We believe that these two aspects must be taken into account to extract relevant dialogue patterns: human dialogue is multimodal and multifunctional. To that end, we propose a two-dimensional data structure called *annotation*

Speaker	Raw dialogues	Annotations		
M	Good morning sir, what can I do for you? <smiling> <looking at the customer>	E	J	V
C	I'd like to buy some black tea. <mutual eye contact>	A	-	M
C	D'you have any special tea for Christmas?	D	-	-
M	Yes we have one, let me check... <checking a tea caddy>	E	-	I
M	I'm afraid we don't have any more in stock. <looking sorry>	A	Sa	I
C	What?	E	Su	-
M	However, we'll be restocked this afternoon <looking at the customer>	A	-	V
C	That's perfect! <pause> I'll be back tomorrow.	A	J	-

Table 1: Annotations of a fictitious dialogue between a merchant and a customer, respectively denoted by M and C in the first column. The first annotation column encodes the fictitious dialogue act of the utterance[36] (E: engaging, A: assertive, D: directive). The second column is dedicated to emotions[37] (J: Joy, Sa: sadness, Su: surprise) and the last column to the gazes (M: mutual eye contact, V: unilateral gaze of the speaker, I: gaze to an other identified element). The character '-' is used when no annotation appear in a given column for a given utterance.

array in which the lines are associated to the dialogue utterances ordered chronologically and the columns are associated to relevant functions and modalities.

The choice of the annotation scheme depends on the application. For example, if a corpus composed of police interrogation dialogues is considered, it is likely that one or several columns of the scheme contain annotations dedicated to the presence of a question or an answer in the utterances. Hence, the difficulty to automatized the creation of the annotations directly depends on the annotation scheme considered. Table 1 represents an annotation array associated to a fictitious dialogue. This example highlight the particularity of the annotation array whose lines (i.e., utterances) are ordered chronologically whereas the order of the column is arbitrary. This data structure is very specific and, to the best of our knowledge, there is no existing method dedicated to the extraction of regularities in a corpus of annotation arrays.

In this context, we propose a two-step approach to ease the identification of regularities in a corpus of annotated dialogues. In the first step, recurrent dialogue patterns are extracted via a dynamic programming algorithm. They are then clustered in a second step. To illustrate our approach, we select a corpus of annotated dialogues between a child and a parent in an interactive storytelling context.

This article is organized as follows. The next section is dedicated to a state of the art on regularity extraction in various fields. The representation of the data and the two-step method are described in Section 3. In Section 4 the experiment and the corpus considered are detailed. Eventually, prior the conclusion in Section 6, the results are presented in Section 5.

2 Related work

With regard to dialogue systems and models, several approaches exist. The *finite-state* approach represents the structure of the dialogue as a finite-state automaton where each utterance leads to a new state[28]. In practice, this approach is limited to system-directed dialogues. The *frame-based* approach models the dialogue as a process of filling in a form containing slots[4]. Unfortunately, the possible contributions of the system are fixed in advance. The *plan-based* approach[2] combines plan recognition with the Speech Act theory[36]. This last approach is rather complex from a computational perspective and requires advanced NLU components in order to infer the speaker's intentions. The *logic-based* approach represents the dialogue and its context in some logical formalism and takes advantage of mechanisms such as dialogue games[27] or inference[20]. Most of the logic based approach works are only on a theoretical level. Finally, the *machine learning* approach proposes techniques such

as reinforcement learning[14] to model the dialogue with Markov Decision Processes. This approach requires an extensive effort of annotation since a very large amount of annotated data is necessary.

Most of these approaches are, partially or entirely, based on the existence of recurrent dialogue patterns occurring in interaction[13]. Hence, we need to identify algorithms which could enable to extract regularities in two-dimensional dialogue annotations. To our knowledge this exact problem has never been tackled previously. This is mainly due to the structure of annotations in which the order of the utterances is important (it corresponds to the chronology of the dialogue) whereas the order of the columns (the annotations) does not matter. In the remaining of this section we review approaches from the litterature used to extract regularities from textual and non-textual data and stress how suitable they are for our problem.

2.1 Regularity extraction on textual data

Regularity extraction from natural language texts is a problem which has already been tackled in several domains. In the field of information extraction, an increasing number of systems are based on a dictionary of linguistic patterns. Such a dictionary is domain specific and time consuming to construct manually. Therefore, several methods have been developped to automatically highlight relevant extraction patterns from a corpus.

One of the earliest system is AutoSlog[35]. It has been used during the Message Understanding Competition 4 (MUC-4) and achieved impressive results by reaching 98% of the performance of a handcrafted dictionary. Through several heuristics, the system specializes a set of syntactic patterns such as *<subject> passive-verb* to match relevant sentences of the corpus. For example during the MUC-4, where articles about terrorism are considered, the previous pattern is specialized into *<victim> was murdered*. However, to extract rules, AutoSlog needs a domain-specific tagging of relevant text parts which can be time consuming to design. To overcome this problem, an adaptation called AutoSlog-TS[34] have been developed. In this version, each text only needs to be labeled as *relevant* or *irrelevant*.

Crystal[40] is a similar system which enables to extract more complicated patterns. Indeed, the mechanisms used to match a given rule are more sophisticated than in AutoSlog where the trigger is nothing more than a single word associated to a context. Subsequently, Crystal rules are iteratively merged as long as they do not produce extraction errors. This process enables to generalize efficiently the rules and to minimize the number of entries in the dictionary. Unfortunately, Crystal requires a semantic tagging of the corpus and a semantic hierarchy of the lexicon which can only be obtained via in-depth knowledge of the corpus.

Yangarber et al[46] propose another approach in which the patterns are incrementally discovered from a small seed set provided by the user. The algorithm iteratively extracts candidate patterns from the corpus and then tries to generalize them. This method enables to minimize the prerequisites on the corpus since only a small number of seeds are required. However, to avoid an exponential processing time, the size and the shape of the patterns are limited.

Sudo et al[41] tackle the corresponding problem in Japanese non-annotated texts. Japanese language raises several specific challenges. Not least among them is the word order flexibility. A given predicate arguments may lead to several patterns. The following example (presented by Sudo et al[41]): "<subj><dobj><iobj><predicate >", with the constraint that the predicate is at the end of the sentence, leads to six possible patterns. To deal with the specific constraints of the language, every relevant sentence is represented by a tree-based pattern. Each node of a tree-based pattern corresponds to a phrasal unit (*e.g.*: verb, subject, company, ...) and each edge denotes a dependency between two nodes. According to the scenario considered, specific phrasal units are added. For example in the case of the Robbery Arrest scenario, the phrasal units *DATE*, *SUSPICION* and *SUSPECT* are introduced. A pattern corresponds to a path in a tree-based pattern and its relevance is determined by its capacity to match the scenario specific phrasal units in the relevant sentences.

A drawback of the above-mentioned information extraction methods is that each of their patterns only covers one sentence. The scope of the patterns is then significantly restricted. Therefore, regularities which last over more than a sentence (*e.g.* "<question><answer>") cannot be extracted with these techniques. In our context these methods are not suitable as they would fail at identifying dialogue patterns covering several speakers.

D'Mello et al[11] consider dialogues between student and tutors. Fifty-one tutoring sessions have been manually transcribed into dialogue moves. A move can either correspond to a speech act or an action. The moves of the tutor have been coded according to a scheme of 27 categories (*e.g.*: direct instruction, explanation, hint, ...). A scheme of 16 moves has been defined for the student (*e.g.*: correct answer, common ground question, ...). A three-step method is then used to extract regularities in the moves. First, the most significant transitions between two moves are detected thanks to a likelihood metric. The results are then represented in a directed graph in which each node is a move and each edge corresponds to one of the identified recurrent transitions. Eventually, graph algorithms are used to extract patterns in the form of paths, cycles and circuits.

Although most of the these mentioned methods give promising results, their mechanisms are specific to text analysis. They rely on the sentence structure which does not exist in two-dimensional annotations. Thus, their adaptation to our problem does not seem to be possible.

2.2 Regularity extraction on non-textual data

Many efforts are dedicated to speech regularity extraction. Barzilay et al[5] aim at identifying speakers in radio broadcasts. To that end, the authors identify several features in order to characterize the speaker. Journalists for example, tend to use characteristic phrases such as *"This is BBC Radio"*. These patterns have been identified thanks to a machine learning method by considering all n-grams from the training corpus for $n \in \{1, 2, 3, 4, 5\}$. The regularity extraction leads to high accuracy identification of the speaker roles. However, the adaptation of this method to corpora of free dialogues in which the speakers do not regularly use the exact same sentence would not be as efficient.

Rao et al[33] consider the problem of designing large digital systems such as compilers. Their aim is to reduce the production cost of the system by reducing it to a collection of subsystems called *templates*. The objective is twofold since both the number of different templates and the total number of templates used to represent the system have to be minimized. The templates are extracted thanks to several heuristics including a greedy algorithm. The purposes of this method are quite different from ours. First, we do not want to minimize the number of patterns used. Furthermore, the patterns sought do not aim to cover all the input dialogues.

Recurrent pattern extraction is a redundant task in data-mining. As stated by Han et al[18], three kinds of pattern can be considered in that field, namely: itemsets, sequences of itemsets and structural patterns (such as sub-graphs and sub-trees). The dialogue representation that we are considering could be seen as a sequence of itemsets. However, the corresponding algorithms are unpractical in our context for two reasons. First, a given itemset $\mathcal{I}$ cannot partially be in a pattern. Either all the items of $\mathcal{I}$ or none are in the pattern. The shape of the patterns is, therefore, strongly constrained. The second drawback is that the proximity of the itemsets in a sequential pattern is not considered. Most of the time in a conversation, the interlocutors react to something said recently. As a result, we assume that time has to be taken into account during the extraction of dialogue patterns. If the two first categories of patterns considered in data-mining (itemsets and sequences of itemsets) do not suit two-dimensional annotations, structural patterns however could do. Whenever a database contains structural information (*i.e.:* relations between its elements), the identification of structural patterns can be performed. Even if most of the efforts are dedicated to exact pattern extraction, the SubDue system[19] seeks approximate structural patterns. This method uses a graph representation in which patterns - initially defined as one node - are iteratively extended by one neighboring edge. The purpose of this system is both to reduce the complexity and the size of the data. Within this scope, the method uses the minimum description length (MDL) principle and therefore tends to highlight the largest patterns instead of the most recurrent. On top of that, the graph required to represent an annotation array would be almost complete which would lead to prohibitive computation times.

None of the previously presented approaches from the litterature are suitable to tackle the problem of extracting dialogue patterns from two-dimensional annotations. We therefore develop an ad hoc method described more thoroughly in the next section.

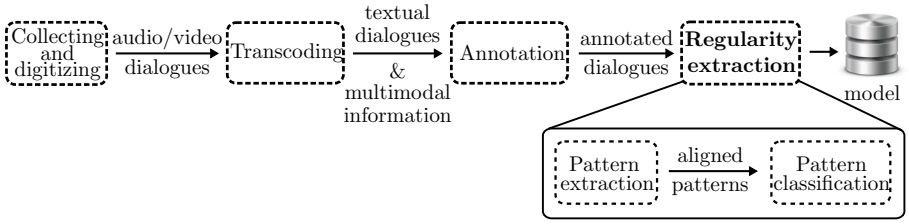


Figure 1: Workflow of the methodology considered to obtain a robust dialogue model. The two-step approach proposed in this paper to ease the extraction of dialogue patterns is represented at the bottom right of the figure.

3 Model

3.1 Overview

Our work falls within the framework represented in Figure 1 which aims at building a robust dialogue model.

Under this approach, textual dialogues - composed of the dialogue transcription and potentially multimodal information are first obtained via a transcoding step. The two first columns of Table 1 represent a fictitious example of such a transcoding in which the multimodal information are represented between diples.

The dialogue representation is then completed by additional information during the annotation step. Each utterance is characterized by an annotation vector whose components match different coding dimensions. Consequently, a dialogue is represented by a two-dimensional table: one row by utterance, one column by coding dimension. Each dimension holds its own annotation alphabet.

A given annotation can occur from several multimodal information. For example in Table 1, annotation J appears in two different utterances, firstly, thanks to the facial expression "*smiling*" and then via the sentence "*That's perfect!*".

The annotation step can be carried out manually or automatically[1]. Eventually, in the last step of the workflow, regularities are extracted and used as a reliable basis to create a dialogue model. For example, Dubuisson Duplessis et al[12] extract dialogue games directly from dialogues annotated thanks to the Dit++ annotation scheme[6].

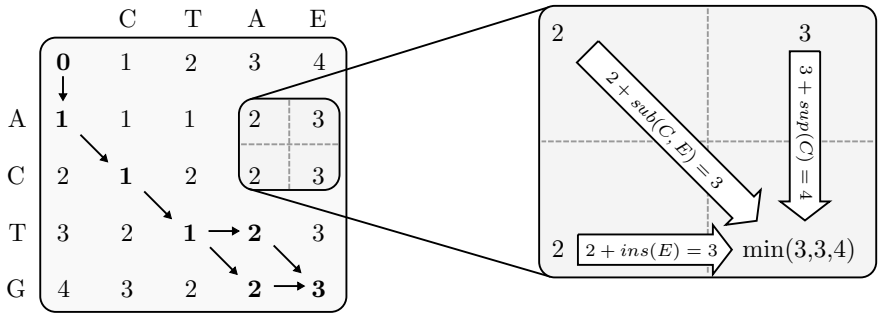
The aim of our work is to guide the identification of regularities from a corpus of annotation arrays. For this purpose, we design a two-step approach (see Figure 1). We first collect recurrent patterns from annotation arrays of the corpus. This task is achieved thanks to the extraction of two-dimensional *local alignments* from every possible pair of dialogues. A two-dimensional local alignment denotes two similar sub-parts in a pair of dialogue annotations. Each of these sub-parts is considered as a pattern. To extract the alignments, we adapted a method called LPCA[25] and reduced its complexity. Each of the extracted pattern occurs at least twice in the corpus. Nevertheless, these two occurrences may by due to sheer coincidence and this is not sufficient to consider the pattern as a regularity. As a consequence, a clustering step is then performed to highlight the most significant patterns. For this purpose, several clustering heuristics have been implemented. In the following of this section, these two steps and the adaptation of LPCA in particular are further detailed.

3.2 Pattern extraction

Let $\mathcal{C} = \{d_i\}_{i=1}^{n}$ be a corpus of n annotations of dialogues. To extract recurrent patterns from $\mathcal{C}$ we identify local alignments from all pairs (d_i, d_j), $i, j \in [\![1, n]\!]$. An alignment can be defined as a pair of (s_i, s_j) such that $s_i \subseteq d_i$ and $s_j \subseteq d_j$ and whose similarity, according to a given metric, is significant (higher than a given threshold). An alignment is said to be *global* if $s_i = d_i$ and $s_j = d_j$; otherwise it is called *local*. Global alignments are to restrictive in our context, as the obtained patterns would necessarily cover the whole dialogue. As a consequence we are interested in obtaining local alignments.

To find local alignments, the challenge consists in defining a proper similarity function and a corresponding algorithm to efficiently compute it. Depending on the time constraint and the size of the considered annotations of dialogues, the algorithm can either return optimal or approximate solutions.

Pattern extraction is extensively used in biology, where the elements are DNA sequences and the purpose is to find similar sub-sequences which can suggest a link between differ-



(a) Table T. The arcs represent the paths followed by the algorithm to obtain the two optimal global alignments represented in Figure 2b. The three terms of the recurrence formula are illustrated by a zoom on a subpart of the table.

$$\begin{pmatrix} A & C & T & G & - \\ - & C & T & A & T \end{pmatrix} \begin{pmatrix} A & C & T & - & G \\ - & C & T & A & T \end{pmatrix}$$

(b) Global alignments obtained.

Figure 2: Global alignments of sequences "ACTG" and "CTAE" with Needleman-Wunsch algorithm.

ent species. This is closely related to our problematic since the annotations of a dialogue can be represented as a set of juxtaposed columns, in which each column is a sequence of annotations. Numerous algorithms have been developed to align DNA sequences[3, 31, 32]. Approximate methods such as BLAST[3] and FASTA[31] enable to quickly align a given sequence with all the elements of a DNA database. Among the exact methods, a similarity function derived from the *Levenshtein distance* is often used.

In the remaining of this section, we first describe a dynamic programming algorithm which returns optimal local alignments for the Levenshtein distance. We thereafter present a generalization called LPCA[25] which takes two-dimensional inputs. Eventually, we describe how we adapt LPCA to reduce its worst-case complexity and improve its trace-back step when extracting local alignments from annotations of dialogues.

3.2.1 Sequence local alignments

Given two sequences of characters e_1 and e_2, of size m_1 and m_2 respectively, the *Levenshtein distance* $ed(e_1, e_2)$ is equal to the minimum number of edit operations (insertion, deletion and substitution) to transform e_1 into e_2. To compute the Levenshtein distance, a cost of 1 is assigned to each edit operation except for the substitution of a character with itself which costs 0. More generally, when each edit operation has its own cost the term *edit distance* is used instead of Levenshtein distance. According to this definition, an alignment corresponds to a superposition of e_1 and e_2 with potential gaps in them.

Needleman-Wunsch algorithm[29] is a dynamic programming algorithm which can efficiently compute the edit distance.

Firstly, the algorithm produces a table T of size $(m_1+1)\times(m_2+1)$ such that $T[i][j]$ equals the edit distance between $e_1[1..i]$ and $e_2[1..j]$, for all (i, j) in $\{1, \ldots, m_1\} \times \{1, \ldots, m_2\}$. As a result, the edit distance between e_1 and e_2 is equal to $T[m_1][m_2]$. The basic idea of this algorithm is that the last edit operation of the optimal alignment can either be a substitution, an insertion or a deletion.

For instance, the edit distance between sequences "ATCA" and "ATGC" can be formulated as

$$ed(ATGC, ATCA) = min \begin{cases} ed(ATG, ATC) & + & sub(C, A) \\ ed(ATG, ATCA) & + & del(C) \\ ed(ATGC, ATC) & + & ins(A) \end{cases} . \quad (1)$$

This leads to a recurrence formula with three terms to minimize. This formula is highlighted in Figure 2a which represents the table T obtained when the sequences "ACTG" and "CTAE" are aligned. Moreover, the table T obtained when computing the Levenshtein distance on the example used in equation (1) is presented in Figure 3 (A).

In a second step, a trace-back from $T[m_1][m_2]$ to $T[0][0]$ which infers the optimal alignment is performed. If several paths lead to $T[m_1][m_2]$, each of them gives a different align-

		A	T	G	C
	A	0	1	2	3
	T	1	0	1	2
	C	2	1	1	1
	A	3	2	2	2

(a) Table T (without its first column and row).

$$\begin{pmatrix} A & T & G & C \\ A & T & C & A \end{pmatrix}$$

$$\begin{pmatrix} A & T & G & C & - \\ A & T & - & C & A \end{pmatrix}$$

(b) Corresponding optimal global alignments.

Figure 3: Results of Needleman-Wunsch algorithm, with the costs of the Levenshtein distance, applied on the two sequences "ATCA" and "ATGC".

ment (see example in Figure 2). These alignments are global since both e_1 and e_2 characters are exhaustively contained in them.

To compute local sequence alignments, Smith-Waterman algorithm[39], a variation of Needleman-Wunsch algorithm, can be used. Therefore, the three edit operation costs are replaced by positive and negative integer scores. Furthermore the values of T are truncated to 0 and now correspond to local similarities instead of global distances. Eventually, the optimal local alignment is obtained using a trace-back from the position with the highest score of T - rather than from (m_1, m_2) - and proceeds until a position with a score of 0 is encountered.

3.2.2 Two-dimensional array local alignment

Alignment extraction from inputs of more than one dimension is not trivial. The challenge relies on defining edit operations which can exhaustively cover all possible alignment shapes. An adaptation of Smith-Waterman algorithm to two-dimensional inputs has recently been developed[25]. The main drawback of this method is that, due to the directions considered in the dynamic programming algorithm, a few pattern shapes cannot be detected. Nevertheless, the fact that these cases are marginal and that the algorithm has a relatively low complexity ($O(n^4)$) makes it the most suitable candidate, to our knowledge, for recurrent pattern extraction in two-dimensional arrays.

Given two two-dimensional arrays d_1 and d_2 of respective size $m_1 \times n_1$ and $m_2 \times n_2$, this algorithm computes a four-dimensional table T of size $(m_1+1)\times(n_1+1)\times(m_2+1)\times(n_2+1)$ such that $T[i][j][k][l]$ is equal to the local similarity between $d_1[1..i][1..j]$ and $d_2[1..k][1..l]$ for all $(i,j,k,l) \in \{1,\dots,m_1\} \times \{1,\dots,n_1\} \times \{1,\dots,m_2\} \times \{1,\dots n_2\}$. To compute T, two four dimensional tables R and C, of the same size than T, are calculated thanks to the Smith-Waterman algorithm such that :

- $R[i][j][k][l]$ equals the local similarity between $d_1[i][0..j]$ and $d_2[k][0..l]$;
- $C[i][j][k][l]$ equals the local similarity between $d_1[0..i][j]$ and $d_2[0..k][l]$.

Table T is then computed by considering, eight two-dimensional edit operations (represented in Figure 4). For a given position $[i,j,k,l]$ in T let:

- $R[i][j][k][l]$ and $R[i-1][j][k-1][l]$ be respectively denoted by r and r';
- $C[i][j][k][l]$ and $C[i][j-1][k][l-1]$ be respectively referred to as c and c';
- $\forall x \in \mathbb{R},\ q(x) = \begin{cases} x & \text{if } x \neq 0 \\ del(d_1[i][j]) + ins(d_2[k][l]) & \text{otherwise} \end{cases}$.

The recurrence formula used to compute table T can be written as

$$T[i,j,k,l] = \max \begin{cases} T[i-1,j,k,l] + ins(d_1[i][j]) & (a) \\ T[i,j-1,k,l] + ins(d_1[i][j]) & (b) \\ T[i,j,k-1,l] + del(d_2[k][l]) & (c) \\ T[i,j,k,l-1] + del(d_2[k][l]) & (d) \\ T[i-1,j,k-1,l] + q(r) & (e) \\ T[i,j-1,k,l-1] + q(c) & (f) \\ T[i-1,j-1,k-1,l-1] + q(r'+c) & (g) \\ T[i-1,j-1,k-1,l-1] + q(r+c') & (h) \\ 0 \end{cases}.$$

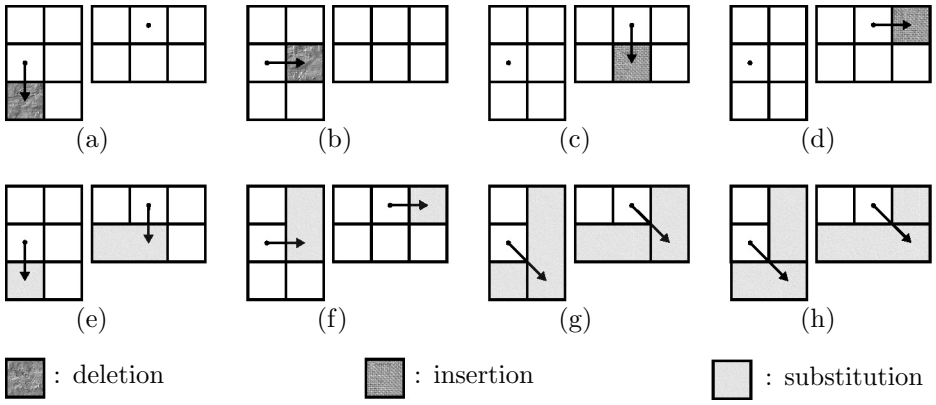


Figure 4: Two-dimensional edit operations considered on two arrays of respective size 2×3 and 3×2. The current position of the algorithm in both arrays is represented by dots and the characters involved by the edit operation are highlighted in gray. **(a)**: Row deletion. **(b)**: Column deletion. **(c)**: Row insertion. **(d)**: Column insertion. **(e)**: Row substitution. **(f)**: Column substitution. **(g)** and **(h)**: Column and row substitutions (in different orders).

Each line of the previous system corresponds to a direction of the dynamic programming algorithm represented in Figure 4. The last line allows to start a new alignment in position (i, j, k, l) whenever the scores of the eight other directions are negative. For a further description of the formula, the reader may refer to Lecroq et al[25].

This algorithm is promising but has two main drawbacks in our context:

- the computation of T on all possible pairs of dialogue annotations is significant whenever the sizes of both the corpus and the dialogues increase;
- the trace-back algorithm only returns the best local alignments. However, two dialogues may have more than one common subpart and we would like to obtain one alignment for each of them.

In the following we present how we adapt LPCA to suits our requirements.

3.3 Alignment of dialogue annotations

LPCA is designed to extract alignments from any data represented as a two-dimensional array. However, the specific structure of dialogue annotations can be exploited to reduce the execution time.

As previously shown in Table 1, each annotation column corresponds to an independent coding dimension with its own alphabet. Therefore, the alignment of two annotations from different columns is irrelevant. Consequently, the column insertion and the column deletion (edit operations (b) and (d) in Figure 4) can be removed. This restriction enables to decrease by two the number of terms in the recurrence formula. Consequently, an annotation is always aligned with another one in the same column. Dimensions two and four of T are therefore redundant. As a consequence, the dimension of table T can be reduced to three and $T[i][j][k]$ corresponds to the local similarity between $d_1[1..i][1..j]$ and $d_2[1..k][1..j]$. This modification enables to improve the worst-case complexity of the algorithm from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^3)$, thus enabling to significantly reduce its computation time. To evaluate the improvement, we compare the performances of LPCA with those of our adaptation as presented in Table 2. This table presents the mean running time and standard deviations of both algorithms when aligning 400 couples of dialogues which contain from 10 to 100 lines. It clearly illustrates that our adaptation significantly reduces the computation time. This improvement is all the more crucial as the number of couples of annotated arrays to align when considering a corpus of n arrays is $\frac{n(n-1)}{2}$.

Once T is obtained, similarly to the one-dimensional algorithm, a trace-back is performed from the position with the best score to get the optimal local alignment. This is not totally satisfying when extracting patterns from a corpus of annotated dialogues. Indeed, the second adaptation applied to LPCA was the modification of the trace-back step to not only obtain the best local alignment but all the relevant local alignments between the two considered dialogues instead. To this end, the positions of T whose score is above a given threshold τ are first tagged as *candidate* positions. One could think of performing a trace-back from each

Number of lines	Our version m	σ	LPCA m	σ
10	0	1	0	1
20	0	1	1	3
30	0	1	2	4
40	1	1	3	3
50	1	1	4	3
60	2	1	7	4
70	3	1	8	3
80	4	2	13	5
90	5	2	15	6
100	6	2	18	7

Table 2: Mean computation times (m) and standard deviations (σ) in milliseconds obtained with the original and the adapted LPCA algorithm when aligning 400 couples of annotation arrays with different number of lines.

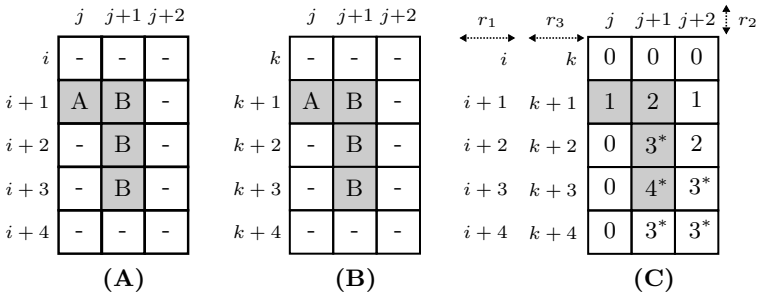


Figure 5: **(A)** and **(B)** Two annotations of dialogues. **(C)** Values of the corresponding table $T[r_1][r_2][r_3]$ within the interval $r_1 \in \{i, \ldots, i+4\}, r_2 \in \{j, \ldots, j+2\}, r_3 \in \{k, \ldots, k+4\}$ and such that $r_3 = r_1 + k - i$. Stars are used to represent the position of T which are candidates if a minimum score of three is considered. The cells in gray represent the locally optimal local alignment and their path in T.

candidate position, however this would lead to a very huge number of redundant alignments. This is exemplified by Figure 5. The two first subfigures (A) and (B) represent similar subparts of two dialogues. In this example, each edit operation has a score of -1 except $sub(A, A)$ and $sub(B, B)$ whose score is equal to 1. Thus, the substitutions of A and the three B lead to a locally optimal score of 4. The path in T followed by the algorithm to reach this score corresponds to the directions related to four substitutions which goes from (i, j, k) to $(i + 3, j + 1, k + 3)$ and can be represented as follows:

$$(i,j,k) \overset{sub(A,A)}{\longrightarrow} (i+1,j,k+1) \overset{sub(B,B)}{\longrightarrow} (i+1,j+1,k+1) \overset{sub(B,B)}{\longrightarrow}$$

$$(i+2,j+1,k+2) \overset{sub(B,B)}{\longrightarrow} (i+3,j+1,k+3).$$

In each of these positions of $T[r_1][r_2][r_3]$, the following relation is satisfied: $r_3 = r_1 + k - i$. This path can thus be represented in a two-dimensional array as shown in subfigure (C). In this context, if the minimum score τ is set to 3, the position with the highest score $(i + 3, j + 1, k + 3)$ is, as expected, a candidate position. However, unexpected positions on the path taken (*e.g.* $(i + 2, j + 1, k + 3)$) and around the path (*e.g.* $(i + 3, j + 2, k + 3)$) are also candidates. Those positions (represented with a star in Figure 5) lead to undesirable alignments which are slight variations of the locally optimal alignment. In the example, only four unexpected positions are highlighted since we only consider the positions such that $r_3 = r_1 + k - i$ and since the optimal score of the alignment is close to τ. In practice, there can be a huge number of irrelevant candidate positions.

To avoid suboptimal alignments, the candidate positions are filtered to only retain the one which are locally optimal in T. Finally, a trace-back is performed from each remaining

Name	Type	Number of clusters
Single-link[38]	Hierarchical	$\{1, \ldots, n\}$
Rock[17]		$\{1, \ldots, n\}$
Chameleon[23]		$\{1, \ldots, c\}$ with $c \leq n$ determined by the algorithm
Spectral clustering[43]	Partitional	Fixed by the user
Affinity propagation[15]		Determined by the method

Table 3: Implemented clustering heuristics and their characteristics. The last column represents the number of clusters k returned by the method. The constant n represents the number of patterns to cluster. Hierarchical methods return a set of partitions with increasing values of k.

candidate positions. These alignments are composed of two sets of annotations (one for each of the considered dialogues) which correspond to two different patterns. For instance in Figure 5, the gray part of (A) is a pattern which has been aligned with the gray pattern of (B). By applying our method on each pair of dialogues, a set of patterns is obtained. This set corresponds to the dialogue annotation sub-parts that occur at least twice in the corpus.

To speed up the trace-back step, a last modification has been made on the original method. During the algorithm, a three dimensional table T_M with T dimensions is filled with numerical values which indicate the directions followed to reach every position. For example in Figure 5, $T_M[i+1][j+1][k+1]$ contains a value which indicates that a column substitution (edit operation (f) in Figure 4) leads to position $(i+1, j+1, k+1)$. As a result, the path followed via the recurrence formula is immediately obtained thanks to T_M, enabling to gain a considerable amount of time with a reasonable counterpart in memory to store T_M.

We subsequently describe how clustering heuristics are used to highlight to the expert the most recurrent patterns.

3.4 Clustering of dialogue patterns

The patterns extracted during the previous step may not all be representative of frequent behaviors since they may appear only in a single alignment. Furthermore, the number of patterns may be too high to directly allow an expert evaluation. We thus carry out a clustering step, with large clusters highlighting the most frequent patterns and small clusters corresponding to marginal patterns.

As pointed out by Jain[21], one of the fundamental questions of the clustering problem is *"How do we define the pair-wise similarity?"*. Here we take advantage of the algorithm described in the previous section and adapt it to compute the similarity of all couples of patterns. Indeed, slight modifications of our extraction algorithm enable to compute a global edition distance instead of a local edition score. It remains to choose a clustering algorithm which takes as input an array of pair-wise similarities.

The clustering problem is well known and extensively studied since the last decades. As a result, numerous clustering techniques have been developed. Since it is not trivial to define the best way of clustering patterns in a given context, we have selected five promising clustering heuristics (see Table 3) based on various mechanisms (projection, neighborhood, connectivity, links and propagation). Two of them (namely: Spectral clustering and the Affinity propagation) return a partition of the patterns, whereas the others are hierarchical and return a nested set of partitions.

At each step of hierarchical methods, the two closest clusters (according to a given similarity) are merged. In Single-link, the similarity of two clusters simply corresponds to the highest similarity between two of their elements. In Rock, the similarity is based on the notion of neighborhood, whereas Chameleon uses two indexes called proximity and the inter-connectivity to compute it.

Spectral clustering stands on the projection of the data in a different space where the clustering itself is performed. Three versions of this method have been implemented, namely : the unnormalized spectral clustering, Shi and Malik's version and Jordan and Weiss' version. These three methods use different Laplacian graphs[43].

Finally, the Affinity propagation is a recent heuristics developed by Frey and Dueck that have the interesting advantage of returning for each cluster a representative which is part

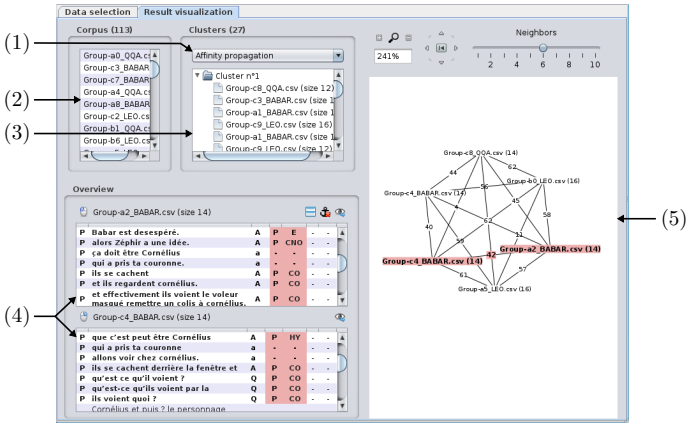


Figure 6: Screen-shot of the application *viesa*. **(1)** : Clustering method selection. **(2)** : Dialogue selection. **(3)** : Pattern selection. **(4)** : Pattern visualization. **(5)** : Cluster visualization.

of the original data. At each step of the algorithm, two kind of messages are exchanged between the patterns, leading to the emergence of clusters with representatives.

The pattern extraction algorithm previously described and the visualization of the different clustering results have been implemented through an ad-hoc software called *viesa* (see screen-shot in Figure 6) available on sourceforge at the following url: *http://sourceforge.net/projects/viesa*.

4 Experimental evaluation

The problem of extracting regularities from a corpus of two-dimensional annotation arrays has not yet been considered in the litterature. As a consequence, we do not have the opportunity to compare our results with the one of other approaches. However, an expert can evaluate the influence of the various parameters of our methodology and assess if it helps the identification of previously unknown regularities.

4.1 Context

The presently described experiment has the dual objective of evaluating both the dialogue pattern extraction step and the clustering step. We consider a corpus composed of 113 dialogues between a parent and his or her child in a storytelling context in which a character is confronted with a false belief[8, 9]. This corpus has been created as a part of a psychological study to identify dialogical characteristics of adult's discourses according to the age of the child they speak to[45]. The number of utterances of the dialogues is between 28 and 249 for a mean value of 82. The corpus has been manually annotated along four columns (see example of dialogue Table 4) :

- The first column is dedicated to the reference of the utterance ((C): to a character of the story, (S): to the speaker or (H): to the interlocutor);
- The second column describes the underlying mental states of the utterance ((V): volition, (OC) or (NOC): observable or non-observable cognition, (A): assumption, (S): surprise, (E): emotion and (K): epistemic statement);
- The two last columns are related to the explanations ((C): by cause - consequence, (O): by opposition, (ES): to explain the story and (PC): to precise the situation through a personal context).

In each column the symbol "-" represents the absence of annotation in a given utterance.

For instance, the utterance line 112 of the example represented in Table 4 is referring to Babar who is a character of the story, which leads to a "C" in the first column. The mental state "*didn't see*" corresponds to an observable cognition, thus the annotation "OC" appears in the second column. Finally "*but*" is here used to introduce a justification by opposition concerning a character's perspective, so the annotations "O" and "ES" are displayed in the two last columns.

Line	Locutor	Utterance	Annotations			
107	Parent	What is it?	-	-	-	-
108	Child	The crown	-	-	-	-
109	Parent	What is it? I don't hear you	-	-	-	-
110	Child	The crown.	-	-	-	-
111	Parent	It's the crown,	-	-	-	-
112	Parent	but Babar didn't see it.	C	OC	O	ES
113	Parent	It is hidden behind the door,	-	OC	-	-
114	Parent	but you see it, it's good.	S	OC	O	PC

Table 4: Excerpt of an annotated dialogue from the storytelling corpus.

Substitution score array	Minimum score τ 36	38	40
$\mathbf{S_-}$	220	124	48
$\mathbf{S_{ref}}$	394	222	120
$\mathbf{S_+}$	754	444	242

Table 5: Number of patterns extracted with respect to the two parameters (namely: the substitution score array and the minimum score τ).

This corpus has been designed as part of a psychological study on interpretations about the behavior of a character who is confronted with a false belief[8, 9]. The aim of this study is to identify semantic and pragmatic characteristics of adult's discourses according to the age of the child they speak to. In this experiment, several sets of patterns are extracted from the corpus and clustered via the selected heuristics. The collected dialogue patterns and the clusters of patterns are evaluated by a psychologist familiar with the corpus and expert in child development of language.

4.2 Pattern extraction parameters

In order to evaluate the pattern extraction step, two parameters have been introduced.

The first parameter is the minimum score τ above which a local alignment is considered to be relevant. This score directly influences the number of extracted patterns. An optimal value of τ, if any, is both data and application dependent. For the experiment three values of τ (namely: 36, 38 and 40) have been heuristically selected. Higher values of τ lead to very small sets of patterns and are therefore likely to omit significant regularities. On the other hand, lower values of τ lead to too many patterns to consider a manual expert evaluation.

The second parameter is the substitution score array. As mentioned in Section 3.2, to extract local alignments, scores have to be assigned to each possible edit operation. The choice of the optimal score for each edit operation has been made by the expert through a trial-and-error method. Consequently, a unique negative score is used for insertions and deletions and a score array S_{ref} is considered for all the possible substitutions. The patterns extracted from the dialogue annotations are directly dependent on S_{ref}. To evaluate its efficiency, two other score arrays S_- and S_+ are considered. In these two arrays, the scores which correspond to the substitution of one annotation by itself are the same than in S_{ref} (it would be irrelevant to penalize with negative scores these edit operations). The other values of S_- and S_+ have been picked pseudo-randomly in order to satisfy the following relation:

$$average(S_-) \leq average(S_{ref}) \leq average(S_+).$$

A set of patterns have been extracted for every possible combination of minimum score and substitution score array. Table 5 presents the number of patterns extracted for each configuration. The table also highlights that the number of patterns decrease as the minimum score and the mean value of the score array increase.

Table 5 also shows that the number of patterns extracted is too high to allow a manual evaluation from an expert. A second step is therefore carried out to cluster each of the nine extracted sets of patterns.

Method	Affinity propagation	Chameleon		Rock			...	Single-link		
Number of clusters	51	5	20	5	20	120	...	5	20	120
Relevance of the clusters	☐	☐	☐	☐	☐	☐	...	☐	☐	☐
Relevance of the number of clusters	☐	☐	☐	☐	☐	☐	...	☐	☐	☐

Table 6: Example of sheet filled during the evaluation for a given minimum score and a given substitution score array. Each blank square was filled with a value between 0 and 4 (0 for "irrelevant" and 4 for "extremely relevant").

4.3 Pattern clustering parameters

The data structures returned by hierarchical and partitional clustering heuristics are different. In the first case, a sequence of nested partitions is obtained whereas in the second case a single partition is given. In this experiment, the comparison of the solutions is possible by examining partitions with close or identical number of clusters. Let k be the number of clusters.

The number of values of k considered for each method increases rapidly the expert's amount of work. Since the evaluation of the solutions is time-consuming, only three values of k - with different orders of magnitude - have been selected for each heuristic, namely: 5, 20 and 120. The number of values considered for k is only restricted in the scope of this experiment in order to highlight the best clustering heuristics. In other cases, the choice of k is left to the user. The three selected values have been chosen after a consultation with the expert. A partition of the patterns in five clusters could enable to highlight regularities with a strong meaning since the average number of patterns by cluster would be high. The choice of the value 20 for k has been made by the expert since it corresponds to the number of previously manunally identified regularities in the corpus. The value 120 corresponds to the maximal number of clusters that are expected in most of the pattern sets considered. Indeed, the patterns are extracted by pairs - since an alignment contains two patterns, thus, k should at the most be equal to the total number of patterns divided by two. As represented in Table 5, most pattern sets contain less than 240 patterns. Therefore, the value 120 is, for them, an upper bound.

Thus, for each hierarchical heuristics, the partitions which correspond to 5, 20 and 120 clusters, if any, are extracted. The expected partitions are obtained from the spectral clustering methods by executing them three times and varying the k parameter. Conversely, the affinity propagation does not give the user the opportunity to fix k. As a result for this method, the value of k is different for each configuration.

In this experiment, two subjective values are considered: the relevance of the number of clusters and the relevance of the clusters. For a given substitution score array and a given minimum score, the expert evaluates these two relevance values for all the clustering solutions obtained.

An example of evaluation sheet is represented in Table 6. One can observe on this example that the method Chameleon does not give a solution for $k = 120$ and that the number of clusters obtained via the Affinity propagation is 51. The partitions returned by the other heuristics for $k = 51$ are not considered in the experiment since we want, as far as possible, to focus on the three values of k previously selected. A fourth value of k (fixed by the result of the Affinity propagation for every pattern set) could have been considered. However, in that case 54 additional partitions (six for each pattern set) would have required an evaluation. Since an expert evaluation of a partition is time consuming, we decided to only consider three values of k for the other heuristics.

5 Results

In this section, the four parameters of the method are evaluated, namely:

- the minimum score τ;
- the similarity matrix;

Clustering method		Rock								...
Minimum score		36			38			40		...
Number of clusters		5	20	120	5	20	120	5	20	120	...
Substitution score array	S_-	0	0	1	0	1	0	0	1	0	...
	S_{ref}	0	2	4	0	1	3	0	1	0	

Table 7: Results of the expert evaluation for the "relevance of the clusters" criteria on S_- and S_{ref} presented as paired measurement.

- the number of clusters;
- the clustering heuristic.

Our aim is to highlight for each of them the optimal value, if any, among those considered during the experiment. To that end, we use a paired difference test called Wilcoxon Signed Rank (WSR) test. The paired difference tests enable to assess whether the means of two paired populations differ. This test has the following hypothesis:

H_0: the median difference between the pairs is zero.

If the test is satisfied, H_0 is rejected and the median difference between the two measurements is therefore significantly different from zero.

As shown in Table 7, given two different values of one parameter, the results of the expert evaluation can be represented as a set of paired measurements. A WSR test can then be performed on all possible pairs of values of each parameter. If a test is satisfied, we deduce that the value of the parameter with the highest mean give significantly better results than the other value. For instance, if we consider the minimum score parameter, which can take the values 36, 38 and 40, we perform a test on each of the three following pairs of values: $(36, 38)$, $(38, 40)$ and $(38, 40)$.

The "relevance of the number of clusters" criteria is used to evaluate the number of clusters whereas the other parameters are evaluated thanks to the index "relevance of the clusters".

The WSR test is considered in this experiment instead of the more commonly used Student's t-test, since the results of the expert cannot be assumed to be normally distributed.

Subsequently to this evaluation, an example of psychologically significant cluster obtained by using the optimal parameters is presented.

5.1 Substitution score array evaluation

In this experiment, three substitution score arrays have been considered, namely: S_{ref}, S_- and S_+. As presented in Section 4.2, the first one has been designed by the expert and the two others were created pseudo-randomly.

The WSR tests are satisfied for (S_-, S_{ref}) and (S_+, S_{ref}). In both cases the mean value of S_{ref} is greater. proving that the substitution score array defined by the expert gives significantly better results than the other two. This could have been expected since more accurate scores help returning more relevant patterns.

In the following of the evaluation, we only kept the results from the expert obtained thanks to S_{ref}.

5.2 Evaluation of the minimum score

The minimum score parameter directly influences the number of patterns extracted. The smaller the value of this parameter, the higher the number of obtained patterns. In this paper, the values 36, 38 and 40 have been considered. These three values were selected heuristically in order to provide a reasonable number of patterns to an expert evaluation. The results of the WSR tests on each pair of values is displayed in Table 8.

The WSR tests for the couples $(36, 38)$ and $(38, 40)$ are both satisfied in favor of the score 38. Thus, this value tends to give better results than the other two values on the considered corpus. Furthermore, the test is not satisfied for the couple $(36, 40)$, so no comparison is possible between these two values. Suprisingly, the minimum score 36, which provides the greater number of patterns does not give the best results. According to feedback from the

	Couple of values		
	$(36, 38)$	$(36, 40)$	$(38, 40)$
Is the WSR test satisfied ?	yes	no	yes
In favor of which value ?	38	-	38

Table 8: Results of the WSR tests on each pairs of values of the minimum score parameter.

	Couple of values		
	$(5, 20)$	$(5, 120)$	$(20, 120)$
Is the WSR test satisfied ?	yes	yes	yes
In favor of which value ?	20	5	20

Table 9: Results of the WSR tests on each pair of considered numbers of cluster.

expert, it seems that too many patterns tends to blur the evaluation by the addition of less relevant patterns.

Thus, the minimum score has a significant impact on the quality of the obtained results. However, the best value of this parameter cannot be determined a priori, as it strongly depends on the substitution scores and the annotation arrays contained in the corpus.

5.3 Evaluation of the number of clusters

The number of clusters is an important parameter, since it directly affects the representation of the data evaluated by the expert. To evaluate its significance three values with different orders of magnitude (namely: 5, 20 and 120) are selected. Table 9 displays the results of the WSR tests for these three values.

The three WSR tests are significant. They show that the best value is 20 and that the worst is 120. As for the minimum score, the intermediate value of the parameter (herein 20) is for the expert a better alternative than the extreme values. According to the expert, the number of dialogical recurrent behaviors covered by the extracted patterns is significantly greater than 5. As a result, when only 5 clusters are required, the clustering algorithms are forced to regroup together unrelated patterns. The obtained clusters are not coherent, which explains their poor quality compared to the one obtained with 20 clusters. On the contrary, 120 clusters appears to be too many. Indeed, it leads to numerous clusters containing only one single dialogue pattern.

Similarly to the minimum score, this parameter greatly influences the quality of the results and its optimal value cannot be initially estimated as it strongly depends on the number of recurrent behaviors contained in the corpus.

5.4 Clustering heuristics evaluation

To evaluate the various clustering heuristics, we first consider the inter-cluster distance. For a given partition, this index is equal to the sum of the distances of all the pairs of patterns which are not in the same cluster. This distance is commonly considered as an objective function in exact clustering approaches[10, 16, 22]. A high inter-cluster distance denotes well separated clusters, which usually implies a relatively good partition. Table 10 presents the values of this index for all the clustering solutions obtained with a minimum score of 38 and the substitution score array S_{ref}. Similar results are observed when varying the value of these two parameters. We can first note that for a given method the inter-cluster distance increases with the number of clusters as expected, since the number of patterns which are not in the same cluster increases. Then, it appears that the normalized spectral clustering methods, the Affinity propagation, Rock and Chameleon tend to return solutions with a high inter-cluster distance. This suggests that the quality of the corresponding solutions are the best. On the other hand, Single-link and the unnormalized spectral clustering heuristics give uneven, and therefore untrustworthy, results.

The inter-cluster distance is one possible indicator to evaluate the quality of clustering solutions. However, we do not know if it constitutes a relevant criteria in the context of our application. Therefore, these results are not sufficient on their own to draw conclusions on the quality of the various clustering heuristics. We now compare these results with

Clustering method	Number of clusters			
	5	20	38	120
Unnormalized spectral clustering	29	100	-	334
Single-link	49	230	-	**358**
Rock	279	341	-	357
Shi and Malik spectral clustering	309	346	-	357
Jordan and Weiss spectral clustering	**312**	**351**	-	**358**
Affinity propagation	-	-	**355**	-
Chameleon	276	-	-	-

Table 10: Rounded values of the inter-cluster distance ($\times 10^{-4}$) for every clustering heuristics according to the number of clusters (solutions obtained with S_{ref} and a minimum score of 38). The symbol '-' is used whenever no solution is produced by the clustering method or if the solution is not considered in the experiment. Bold numbers highlight the best result(s) of each column.

	AP	Ro	SC_u	SC_{sm}	SL	SC_{jw}	Ch
Affinity propagation (AP)		AP	AP	AP	AP	AP	AP
Rock (Ro)	AP		Ro	Ro	Ro	Ro	Ro
Unnormalized spectral clustering (SC_u)	AP	Ro		SC_u	SC_u	-	SC_u
Shi and Malik SC (SC_{sm})	AP	Ro	SC_u		SC_{sm}	SC_{sm}	SC_{sm}
Single-link (SL)	AP	Ro	SC_u	SC_{sm}		SL	SL
Jordan and Weiss SC (SC_{jw})	AP	Ro	-	SC_{sm}	SL		×
Chameleon (Ch)	AP	Ro	SC_u	SC_{sm}	SL	×	

Table 11: Results of the WSR tests on each pair of clustering heuristics, applied on the expert's evaluation. If a test is satisfied, the name of the clustering heuristic which returns significantly better results is displayed. The symbol "×" is used if the test is not satisfied and the character '-' denote that the test cannot be applied.

those obtained with the expert's evaluation by applying WSR tests on all possible pairs of clustering heuristics. The corresponding results are presented in Table 11. For each couple of clustering method, this table represents the name of the one the test favors. It is not possible to compare SC_u and SC_{jw} as their results are often identical. According to the table the Affinity propagation and Rock give the best results. On the other hand Single-link, Chameleon and Jordan and Weiss spectral clustering induce the worst partitions. This can be explained by the fact that these three methods tend to create one very huge cluster and many others reduced to only one pattern which does not allow relevant behavioral conclusions. Moreover, the clusters should at least contain two nodes, as the patterns are extracted in pairs from alignments. The fact that Jordan and Weiss spectral clustering provides on the one hand a high value of the inter-cluster distance index and on the other hand semantically poor clusters implies that the inter-cluster distance criteria is not the most suitable in this context and confirms that the definition of relevant objective functions for the clustering problem of dialogue patterns is not a trivial task. To sum up, this experiment illustrates that the most suitable methods for our clustering problem are the Affinity propagation and Rock. The WSR test between them shows that the former gives better results and therefore it should be preferred. However, the number of clusters obtained when using the Affinity propagation cannot be fixed, and we have seen in Section 5.3 that it is a parameter which directly influence the quality of the results. Thus, whenever the solution from the Affinity propagation is not fully satisfying, Rock should be considered.

5.5 Example of significant cluster

During this experiment, semantically relevant and previously unknown regularities have been extracted from the corpus. As an example, we can consider the three dialogue patterns represented in Table 12 which have been extracted by our dynamic programming algorithm and clustered together via Rock. According to the expert, this cluster represents a strategy used by the parents to provide a mentalist explanation. To express the feelings of a character, the corresponding mental state is first expressed (annotations E or V). It is then explained by two justifications the first, of which contains a close or identical mental state (*e.g.*: "*he*

Speaker	Raw dialogues	Annotations			
P	He is crying.	P	E	-	-
P	Because he is angry.	P	E	C	CH
P	Because he is not happy.	P	E	C	CH

Speaker	Raw dialogues	Annotations			
P	Leo wants Thimothee's shovel.	P	V	-	-
P	But Thimothee doesn't want to give it to him.	P	V	O	CH
P	Because he doesn't have a shovel.	P	-	C	CH

Speaker	Raw dialogues	Annotations			
P	Oh he is not happy.	P	E	-	-
P	Because his friend does not want to give him the shovel.	P	V	C	CH
P	So he gives a big kick in the friend's castle!	P	-	C	CH

Table 12: Three dialogue patterns extracted which have been clustered together by Rock.

*is **crying***", "*because he is **angry***").

This illustrates how our approach helps in identifying dialogue patterns that previous manual extractions failed at obtaining.

6 Conclusion

We describe in this paper a two-step method designed to help the extraction of regularities from two-dimensional annotations of dialogues. Pattern alignments are first extracted from all the pairs of annotated dialogues in a corpus, thanks to a dynamic programming algorithm. The approach presented is heuristic, due to the complexity of the problem and the size of the data under consideration. In the second step, the obtained patterns are clustered via clustering algorithms.

The method provides promising results by highlighting previously unknown and interpretable patterns. Experimental results show that, among various parameters, the choice of the minimum score of an alignment and the number of clusters have a direct influence on the quality of the results. We have also identified two clustering techniques, namely Rock and Affinity propagation, which provide solutions that are exploitable by an expert.

Two main perspectives could be explored to improve the pattern extraction step. In the presented method, the patterns are identified by pairs which ensures that a pattern occurs, approximately or exactly, at least twice in the corpus. However, a pattern which appears only twice is not likely to be the most relevant. Thus, a filtering step could be carried out after the extraction to keep only the most frequent patterns and therefore to reduce the number of elements to cluster during the second step. Secondly, the shape of the patterns obtained by our algorithm is constrained by the order of the annotation columns. This is an undesirable side effect since the columns are independent. To improve the relevance of the patterns, the method has to be adapted to alleviate this limitation (*e.g.:* by improving the recurrence formula of the dynamic programming algorithm) without reducing, to the extent possible, its speed and efficiency.

Regarding the pattern clustering step, the clustering heuristics quickly provide solutions whose quality is not always satisfying. It would be interesting to compare these solutions to the ones obtained thanks to an exact algorithm.

Eventually, our method could also be applied on other corpora to ensure the relevance of the extracted regularities regardless of the input data. A thorough study of the approach efficiency depending on the size of the data (number of dialogues, number of utterances, number of annotation columns, size of the column alphabet) should also be carried out.

Acknowledgements

We would like to thank Emilie Chanoni psychologist at the University of Rouen Normandy for her thorough evaluation of the corpus. We acknowledge support from the PRIMO

project of Institut National des Sciences Appliquées de Rouen Normandy (INSA Rouen Normandie).

References

[1] Ales, Z., Duplessis, G., Serban, O., Pauchet, A.: A methodology to design human-like embodied conversational agents based on dialogue analysis. In: Human-Agent Interaction Design and Models (HAIDM @ AAMAS). pp. 34–49 (2012)

[2] Allen, J., Perrault, C.: Analyzing intention in utterances. Artificial Intelligence 15(3), 143–178 (1980)

[3] Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J., et al.: Basic local alignment search tool. Journal of Molecular Biology (J. Mol. Biol.) 215(3), 403–410 (1990)

[4] Aust, H., Oerder, M., Seide, F., Steinbiss, V.: The philips automatic train timetable information system. Speech Communication 17(3-4), 249–262 (1995)

[5] Barzilay, R., Collins, M., Hirschberg, J., Wittaker, S.: The rules behind roles: Identifying speaker role in radio broadcasts. In: National Conference on Artificial Intelligence (NCAI). pp. 679–684. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2000)

[6] Bunt, H.: The dit++ taxonomy for functional dialogue markup. In: Towards a Standard Markup Language for Embodied Dialogue Acts @ AAMAS. pp. 13–24 (2009)

[7] Bunt, H.: Multifunctionality in dialogue. Computer Speech and Language 25(2), 222–245 (2011)

[8] Chanoni, E.: Rôle du langage dans le développement de la théorie de l'esprit chez les enfants de 3 à 5 ans : contexte verbal et contexte narratif. Ph.D. thesis, Presse Universitaire de Lille (2004)

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

[10] Chopra, S., Rao, M.: Facets of the k-partition polytope. Discrete Applied Mathematics 61(1), 27–48 (1995)

[11] D'Mello, S., Olney, A., Person, N.: Mining collaborative patterns in tutorial dialogues. Journal of Educational Data Mining (JEDM) 2(1), 1–37 (2010)

[12] Dubuisson Duplessis, G., Chaignaud, N., Kotowicz, J.P., Pauchet, A., Pécuchet, J.P.: Empirical specification of dialogue games for an interactive agent. In: Advances in Practical Applications of Agent and Multi-Agent Systems (PAAMS). vol. LNAI 7879, pp. 49–60 (2013)

[13] Duplessis, G.D., Pauchet, A., Chaignaud, N., Kotowicz, J.P.: A conventional dialogue model based on dialogue patterns. International Journal on Artificial Intelligence Tools 26(01), 1–23 (2017)

[14] Frampton, M., Lemon, O.: Recent research advances in reinforcement learning in spoken dialogue systems. The Knowledge Engineering Review 24(04), 375–408 (2009)

[15] Frey, B.J., Dueck, D.: Mixture modeling by affinity propagation. Advances in neural information processing systems 18, 379–386 (2006)

[16] Grötschel, M., Wakabayashi, Y.: Facets of the clique partitioning polytope. Mathematical Programming 47(1), 367–387 (1990)

[17] Guha, S., Rastogi, R., Shim, K.: Rock: A robust clustering algorithm for categorical attributes. In: Data Engineering, 1999., 15th International Conference on. pp. 512–521. IEEE (1999)

[18] Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery 15(1), 55–86 (2007)

[19] Holder, L., Cook, D., Djoko, S., et al.: Substructure discovery in the subdue system. In: AAAI Workshop on Knowledge Discovery in Databases. pp. 169–180 (1994)

[20] Hulstijn, J.: Dialogue games are recipes for joint action. In: Proceedings of the Forth Workshop on the Semantics and Pragmatics of Dialogue (Gotalog'00). pp. 00–05 (2000)

[21] Jain, A.: Data clustering: 50 years beyond k-means. Pattern Recognition Letters 31(8), 651–666 (2010)

[22] Johnson, E., Mehrotra, A., Nemhauser, G.: Min-cut clustering. Mathematical Programming 62(1), 133–151 (1993)

[23] Karypis, G., Han, E., Kumar, V.: Chameleon: Hierarchical clustering using dynamic modeling. Computer 32(8), 68–75 (1999)

[24] Kopp, S., van Welbergen, H., Yaghoubzadeh, R., Buschmeier, H.: An architecture for fluid real-time conversational agents: integrating incremental output generation and input processing. Journal on Multimodal User Interfaces 8(1), 97–108 (2014)

[25] Lecroq, T., Pauchet, A., Chanoni, É., Solano, G.: Pattern discovery in annotated dialogues using dynamic programming. International Journal of Information and Decision Sciences (IJIDS) 6(6), 603–618 (2012)

[26] Levin, J., Moore, J.: Dialogue-games: Metacommunication structures for natural language interaction. In: Distributed Artificial Intelligence. pp. 385–397. Morgan Kaufmann Publishers Inc. (1988)

[27] Mann, W.C.: Dialogue games: Conventions of human interaction. Argumentation 2(4), 511–532 (1988)

[28] McTear, M.: Spoken dialogue technology: toward the conversational user interface. Springer-Verlag New York Inc (2004)

[29] Needleman, S., Wunsch, C., et al.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Moleculay Biology (J. Mol. Biol.) 48(3), 443–453 (1970)

[30] Orkin, J., Roy, D.: Automatic learning and generation of social behavior from collective human gameplay. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 385–392 (2009)

[31] Pearson, W.R., et al.: Rapid and sensitive sequence comparison with fastp and fasta. Methods in Enzymology 183, 63–98 (1990)

[32] Pollard, D., Bergman, C., Stoye, J., Celniker, S., Eisen, M.: Benchmarking tools for the alignment of functional noncoding dna. BMC Bioinformatics 5(1), 1–17 (2004)

[33] Rao, D., Kurdahi, F.: On clustering for maximal regularity extraction. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 12(8), 1198–1208 (1993)

[34] Riloff, E.: Automatically generating extraction patterns from untagged text. In: National Conference on Artificial Intelligence (NCAI). vol. 13, pp. 1044–1049 (1996)

[35] Riloff, E., et al.: Automatically constructing a dictionary for information extraction tasks. In: National Conference on Artificial Intelligence (NCAI). pp. 811–811. JOHN WILEY & SONS LTD (1993)

[36] Searle, J.: Speech Acts: An Essay in the Philosophy of Language. Cambridge University (1969)

[37] Serban, O., Pauchet, A., Pop, H.: Recognizing emotions in short text. In: Filipe, J., Fred, A. (eds.) 4th International Conference on Agents and Artificial Intelligence (ICAART). vol. 1, pp. 477–480. SciTePress (2012)

[38] Sibson, R.: Slink: an optimally efficient algorithm for the single-link cluster method. The Computer Journal 16(1), 30–34 (1973)

[39] Smith, T., Waterman, M.: Identification of common molecular subsequences. Journal of Molecular Biology (J. Mol. Biol.) 147(1), 195 – 197 (1981)

[40] Soderland, S., Fisher, D., Aseltine, J., Lehnert, W.: Crystal: Inducing a conceptual dictionary. In: Fourteenth Internation Joint Conference on Artificial Intelligence (IJCAI). pp. 1314–1319 (1995)

[41] Sudo, K., Sekine, S., Grishman, R.: Automatic pattern acquisition for japanese information extraction. In: First international conference on Human Language Technology Research. pp. 1–7. Association for Computational Linguistics (2001)

[42] Swartout, W.R., Gratch, J., Jr., R.W.H., Hovy, E.H., Marsella, S., Rickel, J., Traum, D.R.: Toward virtual humans. AI Magazine 27(2), 96–108 (2006)

[43] Von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17(4), 395–416 (2007)

[44] Weizenbaum, J.: Eliza—a computer program for the study of natural language communication between man and machine. Communications of the ACM 9(1), 36–45 (1966)

[45] Wimmer, H., Perner, J.: Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception. Cognition 13(1), 103–128 (1983)

[46] Yangarber, R., Grishman, R., Tapanainen, P., Huttunen, S.: Unsupervised discovery of scenario-level patterns for information extraction. In: Sixth conference on Applied Natural Language Processing. pp. 282–289. Association for Computational Linguistics (2000)