VERBAL DIALOGUE VERSUS WRITTEN DIALOGUE

David Burns, Richard Fallon, Phil Lewis, Vladimir Lovitskii, Stuart Owen

Abstract: Modern technology has moved on and completely changed the way that people can use the telephone or mobile to dialogue with information held on computers. Well developed "written speech analysis" does not work with "verbal speech". The main purpose of our article is, firstly, to highlights the problems and, secondly, to shows the possible ways to solve these problems.

Keywords: data mining, speech recognition, natural language processing

ACM Classification Keywords: I.2.7 Natural Language Processing: Text analysis, speech recognition.

Introduction

There are several problems, which distinguish Verbal Dialogue (VD) from Written Dialogue (WD).

Problem 1. *Sensible VD restricts us to include in our reply to a user enquiry a very restricted number of alternative (3, 4, or maximum 5)* choices, because we should take into account our restricted ability to store information temporarily in our memory [1]. For example, assume the user is searching for *"Security Services Company in UK"*. Any information retrieval engine (e.g. Google) will assumes a maximal matching between enquiry words and documents. In the search result user almost immediately has got 14,600,000 documents and now it is the users' problem to identify the most appropriate one. In the case of VD such an approach is unacceptable even for the first 10 documents.

Problem 2. *Wrong recognition of Verbal Enquiry (VE)* (only *speaker independent* speech recognition is considered). As distinct from WD VD does not have any misspelling problem (we assume that grammar includes only correctly spelled words) because any VE will be converted into a sequence of "correct words". For example, a user said his postcode is: *"PL3 4PX "*. There are a minimum of 4 different possible results of recognition of this VE: (1) *"PL3 4PX "*; (2) *"BL3 4PX"*; (3) *"PL3 4BX"*; (4) *"BL3 4BX"*. The last two results: (3) and (4) represent non existing postcodes and Natural Response (NR) system [2] will simply ask user to repeat postcode. In the case of the second result *"BL3 4PX"* represents existing postcode but wrong result.

Problem 3. Barging (VD interruption) is absolutely natural style of VD. This is a fragment of VD:

NR:	*Where would you like to go?*
User:	*The best beaches, of course.*
NR:	OK, *We can offer you Bahamas beach or Mia…*
User:	*Bahamas, please.*

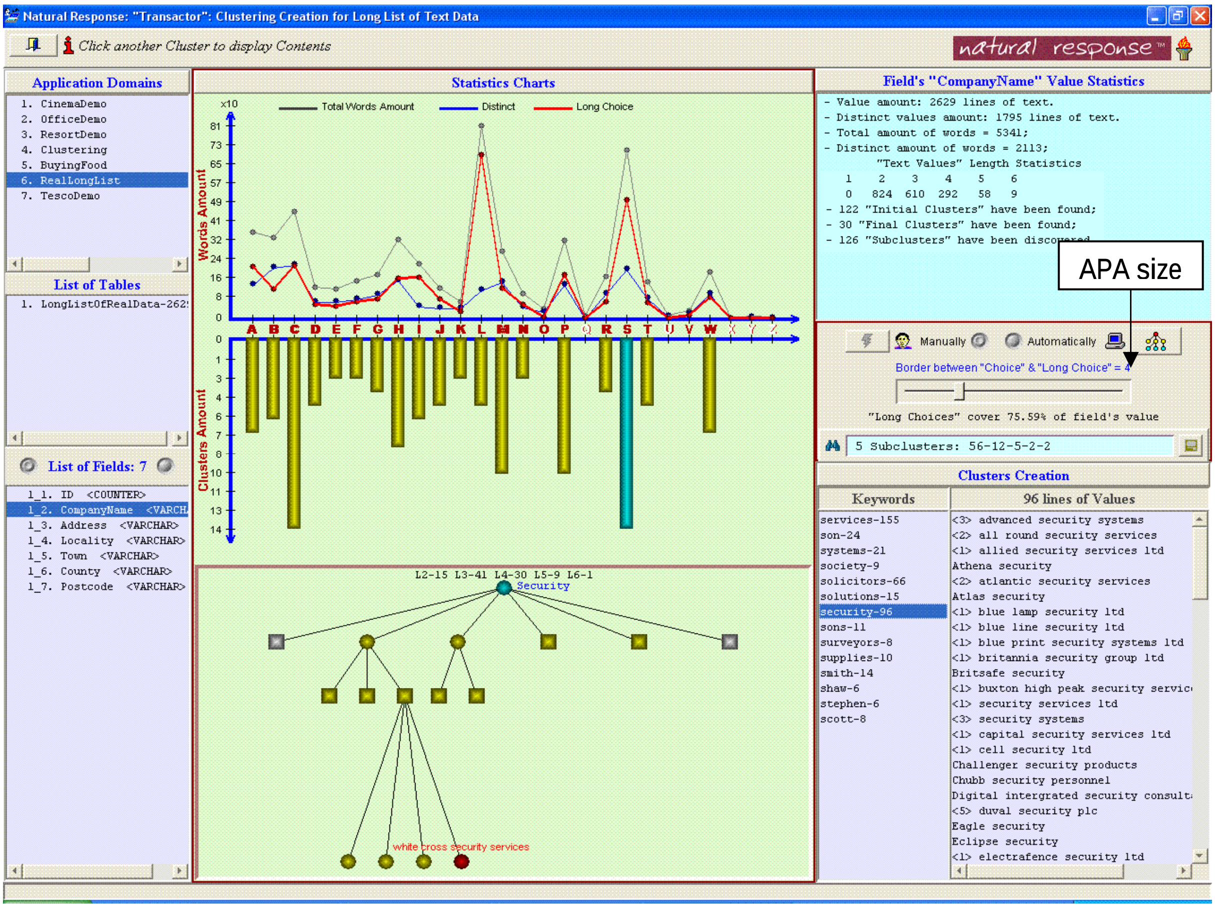
The solution of this problem is more technical than scientific regarding the natural language processing and therefore will not be considered in this paper.

Problem 4. TD verification and mining (TDVM). This problem is not just VD problem but is also a problem of WD. Real customer's DB's very often contain product names like: *"Wht thk slcd brd"* which is intended to mean "*White thick sliced bread"*, or *"Dietary spec gltn/fr & wht/frchoc/orge waf x5"*. A good TDVM analogy is to a body scanner. TDVM software can scan text and identify things that need to be looked at; it will show where there are anomalies in the TD. There is some specific feature of TDVM for VD: TDVM should provide selection of phonetically similar words to try to avoid the use of similar sounding words with different meaning.

Problem 5. Grammars creation. Speaker independent speech recognition has to be 'primed' around a particular vocabulary of terms and phrases, known as Grammars. To achieve reasonable recognition accuracy and response time multiple grammars should be generated automatically as a result of text data (TD) analysis. This overcomes a conflict that exists within speech recognisers that limits the effective vocabulary size without impairing the quality of recognition (normally, quality reduces as vocabulary size increases). Here we mean text data from an Application Domain (AD), which is represented by a database (DB). Let us distinguish two classes of grammars: independent and AD dependent grammars. Examples of independent grammars are *YesNo*, *Date*, *Time*, *Currency*, *CreditCard*, etc. Here we will discuss problem related only to AD dependent grammars creation. A grammar's "accuracy" depends on the solution of the previous problem.

VD: Alternative Choice Minimizing

The Alternative Choice Minimizing Mechanism (ACMM) is involved only when VD problem 1 arises i.e. when in reply to VE the number of "alternative choices" exceeds some pre-specified amount of permissible alternatives (APA). APA depends on the contents of AD values e.g. if values are represented by single words then APA might equal 5: *"What colour do you prefer blue, brown, white, magnolia or crème?"* If values are represented by long sequences of words APA would be better assigned as 3: *"We've got Mothers pride plain medium white bread 800g, Kingsmill square cut medium sliced white loaf 400g, or Hovis classic cut medium white bread 800g. Which do you prefer?"*.



Natural Response: "Transactor": Clustering Creation for Long List of Text Data
Click another Cluster to display Contents
natural response™
Application Domains
List of Tables
List of Fields: 7
Statistics Charts
x10
Total Words Amount
Distinct
Long Choice
Words Amount
81
73
65
57
49
41
32
24
16
8
0
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Clusters Amount
0
1
3
4
6
7
8
10
11
13
14
L2-15 L3-41 L4-30 L5-9 L6-1
Security
white cross security services
Field's "CompanyName" Value Statistics
APA size
Manually
Automatically
Border between "Choice" & "Long Choice" = 4
Clusters Creation
Keywords
96 lines of Values

```
1. CinemaDemo
2. OfficeDemo
3. ResortDemo
4. Clustering
5. BuyingFood
6. RealLongList
7. TescoDemo
```

```
1. LongListOfRealData-262!
```

```
1_1. ID   <COUNTER>
1_2. CompanyName   <VARCH
1_3. Address   <VARCHAR>
1_4. Locality   <VARCHAR>
1_5. Town   <VARCHAR>
1_6. County   <VARCHAR>
1_7. Postcode   <VARCHAR>
```

```
- Value amount: 2629 lines of text.
- Distinct values amount: 1795 lines of text.
- Total amount of words = 5341;
- Distinct amount of words = 2113;
        "Text Values" Length Statistics
   1    2    3    4    5    6
   0   824  610  292   58   9
- 122 "Initial Clusters" have been found;
- 30 "Final Clusters" have been found;
- 126 "Subclusters" have been discovered
```

```
"Long Choices" cover 75.59% of field's value
```

```
5 Subclusters: 56-12-5-2-2
```

```
services-155
son-24
systems-21
society-9
solicitors-66
solutions-15
security-96
sons-11
surveyors-8
supplies-10
smith-14
shaw-6
stephen-6
scott-8
```

```
<3> advanced security systems
<2> all round security services
<1> allied security services ltd
Athena security
<2> atlantic security services
Atlas security
<1> blue lamp security ltd
<1> blue line security ltd
<1> blue print security systems ltd
<1> britannia security group ltd
Britsafe security
<1> buxton high peak security servic
<1> security services ltd
<3> security systems
<1> capital security services ltd
<1> cell security ltd
Challenger security products
Chubb security personnel
Digital intergrated security consult
<5> duval security plc
Eagle security
Eclipse security
<1> electrafence security ltd
```

Figure 1. "Companies-Addresses" Application Domain

The best way to describe the solution to Problem 1 is to use some AD and show step by step the method of the solution. An AD database "Company-Addresses" with 2,629 companies is used (Figure 1). Knowing the TD, the user can assign an appropriate APA size before they start to analyse TD and create clusters.

To show the role of ACMM let us compare two VD.

User: *I'm looking for a security company address.*

NR: *There are 96 different security companies. Could you please be more precise?*

and User: Please help me to find a security company address.

NR: *We found for you Security services, Security systems, Blue security or Security group. Any good for you?*

User: *I'm afraid, not.*

NR: *Ok, what about Security Management, Security solutions, or Reliable security. Is it better?*

User: *Oh, yes, thank you. Security solutions, please.*

NR: *There are Perimeter Security Solutions Ltd. and Security Solutions (Northern) Ltd. Please make your choice.*

User: *Northern.*

NR: *Ok, fine. The address of Security Solutions (Northern) Ltd. is Unit 7/8, Bolton, Lancashire, BL1 8TU.*

If user could not select anything suitable (e.g. instead of select *"Security Solutions"* user said *"No"*) NR will continue dialogue:

NR: *Never mind. Let's try another way. In which city you are looking for Security Company?*

User: *Bolton.*

NR: *We have Blue Print Security Systems Ltd, Cell Security Ltd, Duval Security Plc or Jaymac Security Products Ltd. Any good for you?*

User: *Yes, thank you. I'm looking for Duval Security.*

NR: *Address of Duval Security Plc is 602 Chorley New Road, Bolton, Lancashire, BL6 6EX.*

The difference between 1st and 2nd VD is provided by ACMM. The main features of ACMM are:

- As a result of TD analysis a Pyramidal Structure of Clusters (PRC) is created [3] (fragment of PRC is shown in Figure1). The criterion for each "parent node" of PRC is very simple: "striving for having no more than APA child nodes" (APA = 4). Let us distinguish three possible amounts of cluster/sub-clusters nodes (NA): (1) NA <= APA e.g. for sub-cluster "*Blue Security*" there are just three members {"*Blue Lamp Security Ltd*", "*Blue Line Security Ltd*", "*Blue Print Security Systems Ltd*"}; (2) NA <= APA*APA. Assume there are APA sub-clusters and each of them has no more than APA nodes. In that case replay includes "sub-cluster name" instead of "company name" e.g. "*Security services", "Security systems", "Blue security or "Security group*"; (3) NA > APA*APA.
- In case (1), if the user is looking for "*Blue Security*" NR immediately offers him/her three companies (see above). Case (2) is represented in given VD.
- As for case (3) supplementary field (or fields) (SF) are involved. In the considered VD "*City*" is selected as SF. SF is defined at the stage of Scenario of Dialogue creation. In real AD "Companies-Addresses" except "*City*" SF "*Postcode*" is used.

Wrong Recognition of Verbal Enquiry

Let us consider two different variants of wrong VE recognition. Assume that in the result of TD analysis vocabulary $\mathcal{V}$ of meaningful words and set $\mathcal{S}$ of words sequences (each of which represents a TD line) have been created, and then grammar $\mathcal{G} = \mathcal{V} \cup \mathcal{S}$ has been generated. First of all let us consider the situation when $\mathcal{G} = \mathcal{V}$. In this case when the user said word w_u speech recogniser (SR) using $\mathcal{G}$ produced w_r. Theoretically there are four different results of such event:

(1) $w_u \in \mathcal{V}$ and $w_u = w_r$ (naturally, $w_r \in \mathcal{V}$. But equation $w_u = w_r$ requires some explanation. In the set of recognition results w_i and s_j are distributed in between three intervals in accordance with their scores: "Yes", "Clarifying Question" and "Don't understand" intervals. w_r denotes the top scoring word from "Yes" interval);

(2) $w_u \in \mathcal{V}$ and $w_u \neq w_r$;

(3) $w_u \notin \mathcal{V}$ and $w_u \neq w_r$,

(4) $w_u \notin \mathcal{V}$ and $w_r = nil$ ($w_r = nil$ means that w_r belongs to the “Don't understand” interval).

When $\mathcal{G} = \mathcal{V} \cup \mathcal{S}$ and VE is represented by a sequence of words e.g. $s_u = w_{u1} \rightarrow w_{u2}$ the situation of VE recognition is more complicated. We are not going to discuss here results of recognition for all possible combinations of input VE. We just consider a couple of the most significant cases:

(5) $w_{u1} \in \mathcal{V}, w_{u2} \in \mathcal{V}, s_u \in \mathcal{S}, w_{u1} \neq w_{r1}, w_{u2} \neq w_{r2}$ and $s_u \neq s_r$;

(6) $w_{u1} \in \mathcal{V}, w_{u2} \in \mathcal{V}, s_u \in \mathcal{S}, w_{u1} \neq w_{r1}, w_{u2} \neq w_{r2}$ and $s_r = nil$;

(7) $w_{u1} \notin \mathcal{V}, w_{u2} \notin \mathcal{V}, s_u \notin \mathcal{S}, w_{u1} \neq w_{r1}, w_{u2} \neq w_{r2}$ and $s_u \neq s_r$ (but $s_r \in \mathcal{S}$).

Cases (1) and (4) are obvious. For the solutions of cases (2) and (3) it is better to show on example. Suppose we have AD “Operator”, which is the alternative to a receptionist, providing customers call reception. Assume the next VD took place:

User: *Can I speak to Vlad?* (where $w_u = \mathcal{V}lad$. SR recognised *Vlad* as *Wlad* from “Yes” interval, i.e. $w_r = \mathcal{W}lad$ and $w_r \in \mathcal{V}$ - case (2)).

NR: *Certainly, connecting you now* (i.e. connecting to the wrong person).

<u>Solution</u>. To avoid annoying “clarification questions” e.g. *“Did you say Vlad?”*, recognised word from “Yes” interval is accepted as a “correct word” but such an approach might be cause a wrong result of VD. To sort out this problem a set of heuristics production rules has been developed to check words from the “Yes” interval e.g. “If 1st letter of w_r belongs to $\{b, p, v, w, \ldots\}$ ask *“clarifying question”*.

Case (3) is more complicated. Suppose that the position of someone from the customer's company (AD “Operator”) is not *“Director”* but instead *“Chief Executive Officer”* been used. AD “Operator”, as well as connecting users, can navigate them i.e. describe the direction to the company's office. Another words, “*director*” $\notin \mathcal{V}$ and “*direction*” $\in \mathcal{V}$. The next VD might take place:

User: *Can I speak to a director?* (where w_u = *director*. SR recognised *director* as *direction* from “Yes” interval, i.e. w_r = *direction* and $w_r \in \mathcal{V}$).

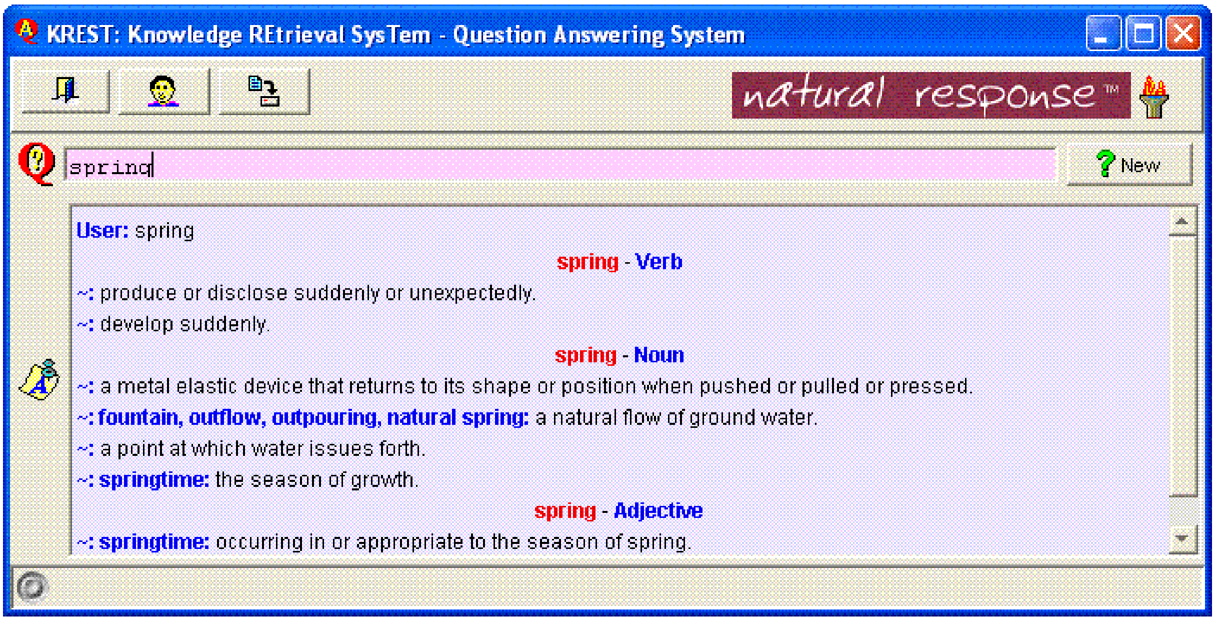
NR: *With pleasure. Where are you now?* (completely wrong VD).

<u>Solution</u>. When AD created it is not enough to include in the knowledge base (KB) of AD only meaningful TD from customer's DB. It is important to extend text values by adding corresponding synonyms. First of all let us distinguish “closed” and “open” TD. For example, in AD “Operator” field “*Name*” represents *closed* data because caller should pronounce a name exactly from the existing list of Names. But field “*Position*” represents *open* data because caller might say: *“I would like to talk with a director (chairman, boss, managing director etc.)”*. In the case of considered field “Position” the solution is very simple because the set of valid positions in UK is well defined and very restricted. It is enough just to provide a link between the existing position i.e. *“Chief Executive Officer”* and set of synonyms i.e. {*director, chairman, boss, managing director etc.*}. Now the previous VD will be changed:

User: *Can I speak to director?*

NR: *Our Company does not have a director position. We have chief executive officer instead. Would you like to speak to him?*

To find an appropriate set of synonyms for an open field is not easy when field represents TD like “*Product*” containing thousands and thousands of different products. On the one hand, it is not difficult to find synonyms for meaningful words of product using, for example, WordNet [6]. WordNet is a lexical database for the English language. Using WordNet's dictionary makes it easy to extract synonyms and meanings for any word e.g. *spring* (see Figure 2). But on the other hand, it is very difficult to automatically select appropriate synonyms for the current AD. To solve this problem the list of VE regarding AD should be analysed.


KREST: Knowledge REtrieval SysTem - Question Answering System
natural response ™
spring
New

User: spring

spring - Verb

~: produce or disclose suddenly or unexpectedly.

~: develop suddenly.

spring - Noun

~: a metal elastic device that returns to its shape or position when pushed or pulled or pressed.

~: **fountain, outflow, outpouring, natural spring:** a natural flow of ground water.

~: a point at which water issues forth.

~: **springtime:** the season of growth.

spring - Adjective

~: **springtime:** occurring in or appropriate to the season of spring.

Figure 2. Synonyms and meaning of word "*spring*"

In the result of VE processing a language model (LM) of VE will be created. LM contains statistical information about which words or word sequences are used mostly by callers of AD. In addition to the statistical information, words adjacent within VE are analysed. The idea of adjacent words is based on the assumption that with the successive presentation of a number of words the strongest relation in it is the relation between the nearest neighbour words. *"Their succeeding one after another presents evidently an important condition of structuring"* [4, p.231]. It is important to underline that this information is always AD dependent. LM is stored in Sequential/Simultaneous Structure (SSS) [5]. LM of VE contains not only a set of synonyms but also patterns of VE. An example of a pattern for AD "Operator" is:

$$(speak \oplus talk \oplus connect) \rightarrow \langle Name\rangle \oplus \langle Position\rangle;$$

where ⊕ - denotes "*exclusive OR*"; → means "*followed by*"; <Name> and <Position> denote any value from DB fields "*Name*" and "*Position*" respectively. Words *speak*, *talk* and *connect* (such words we call "action words") are included in $\mathcal{V}$ along with words from fields "*Name*" and "*Position*". If a pattern of VE is available the initial VD will be have in a more "*natural way*" (even if word *director* is not added to $\mathcal{V}$)**:**

User: *Can I speak to a director?* (SR recognised *director* as *direction* and correctly recognised word *speak* i.e. both *speak* and *direction* have been placed in the same "Yes" interval. Action words have priority and that is why the corresponding VE pattern has been involved to continue VD)

NR: *Do you know the name of who you want to speak to?*

User: *I'm afraid, not.*

NR: *Ok, in that case would you like to talk with chief executive officer?*

User: *Sorry, I change my mind. Can I speak to marketing manager?*

NR: *Certainly, connecting you now.*

To discuss cases (5), (6) and (7) AD "Postcodes", which contains postcodes and cities, will be used. There are 1,741,488 postcodes in the UK. Each Postcode consists of two parts. The first part is the Outward Postcode, or Outcode. There are 2,918 outcodes. Outcode is separated by a single space from the second part, which is the

Inward Postcode, or Incode. There are 4,000 incodes and 27,089 cities and towns in UK. Postcodes recognition is very important because as soon as a postcode has been recognised correctly a very restricted set of full addresses (~30 – 60) is allocated. In order to provide postcode recognition, grammar $\mathcal{G} = \mathcal{V} \cup \mathcal{S}$ must be generated but the size of $\mathcal{G}$ is huge (i.e. $|\mathcal{G}|$ = 1,741,481+2,918+4,000 = 1,748,399 items) and modern SR cannot provide reliable and acceptable recognition for such size of grammar. If we exclude postcodes from $\mathcal{G}$ and add cities it is possible significantly reduce grammar's size i.e. $|\mathcal{G}|$ = 2,918+4,000+27,089 = 34,007 items but increase "intelligence of VD". Below several VD are represented to show the solution of cases (5), (6) and (7).

User: *PL3 4PX.*

NR: *You said PL3 4BX. Is it correct?*

User: *No.*

NR: *Never mind. Let's try another way. Is outcode PL3 correct?*

User: *Yes.*

NR: *Ok, fine. In that case you said PL3 4PX (because only three different incodes with X link to PL3: 4PX, 5PX and 6PX). Is it right?*

User: *Absolutely.*

The next VD is more complicated. Town and city are involved to clarify the wrong result of postcode recognition.

User: *PL11 3BX.*

NR: *It is not right postcode. Did you really say BL11 3BX?*

User: *No.*

NR: *No matter. Let's try another thing. Tell me your town, please.*

User: *Seaton.*

NR: *There are 8 towns Seaton in different areas like Exeter, Sunderland, etc. What is your nearest city?*

User: *Plymouth.*

NR: *Great. The outcode of Seaton in Plymouth area is PL11. Is it correct?*

User: *Of course.*

NR: *Consequently you expressed your postcode as PL11 3BX. Isn't it?*

User: *Oh, at last. I really said so.*

Text Data Verification and Mining

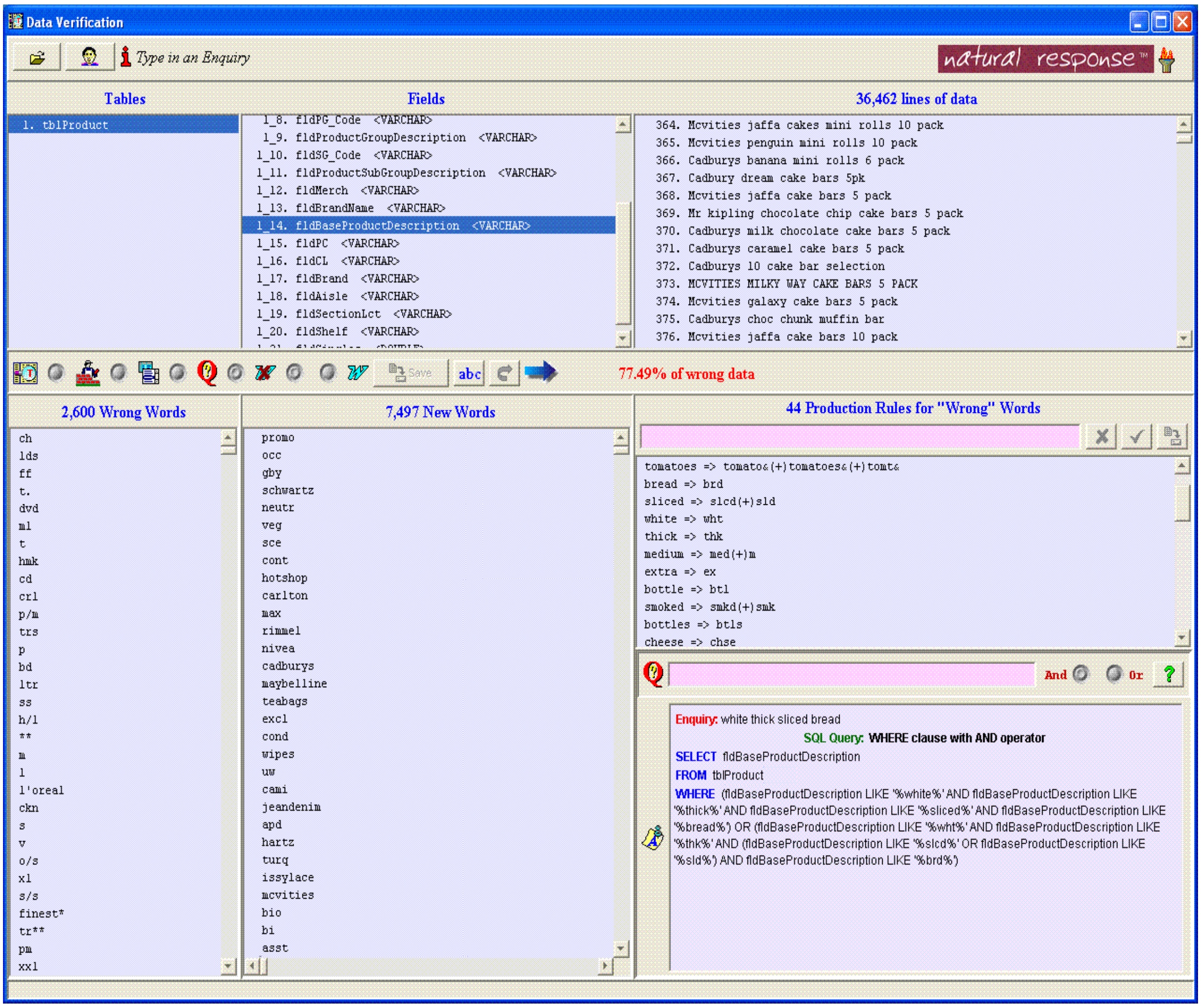
By default two different kind of DB exist in reality – with correct TD and "wrong" TD. For example, DB, which contains TD about *people*, *addresses*, *companies* etc. as a rule have correct TD and require minimum effort for TD verification. Result of TDVM for such TD is shown on Figure 1. The typical example DB with "wrong" TD is any DB with "Product name" in it. Let us consider the real AD "Product" (Figure 3). We will call TD "wrong" if they contain *wrong words* (WW) and/or *unknown words* (UW). Two different kind of WW will be distinguished:

(1) abbreviation*, which is a sequence of consonants letter.* For example, ***"Wht thk slcd brd"*** means "*White thick sliced bread*". (2) Sequence of symbols when sequence includes not only letters but also some symbols e.g. *"Dietary spec gltn/fr & wht/frchoc/orge waf x5"* (We don't know the meaning of that).

The definition of UW is very simple: Let us call word UW if it absent in Oxford dictionary. There are three possible reasons for it: (1) word shortening e.g. "***org"*** for ***organic*** (*org* was used in 261 names of product e.g. "*Doves Farm Org Lemon Cookies Non Gluten"*); (2) new words (= brand) e.g. *Nestle*, *Cadburys* etc.; (3) misspelling. Result of DTVM of AD "Product" is shown on Figure 3.

At the stage of TDVM several tasks need to be solved:

- **Quantity analysis –** count different kinds of frequencies (see Figures 1 and 3);
- **Clustering –** to provide the classification of information into clusters.
- **Data Verification** of TD i.e. extracting WW and UW (see Figure 3). For AD "Product" 76.55% represent wrong data.
- **Production Rules** (PR) **creation**. It is very important to underline the fact that a customer's DB cannot be corrected or reconstituted. *It means that conversion of correct user's enquiry to wrong customer's data needs to be provided.* For such conversions PR need to be created. **Antecedent** of PR might be a single word or sequence of words. Several different WW or UW might represent **consequent** of PR. (+) stands for exclusive OR (see PR on Figure 3).
- **Data Reconstitution** needs to be provided at the step of KB creation when the link between TD and DB Field is made. At this stage each product description is converted using PR from wrong TD to correct TD.



```
1. tblProduct
```

```
 1_8. fldPG_Code  <VARCHAR>
 1_9. fldProductGroupDescription  <VARCHAR>
1_10. fldSG_Code  <VARCHAR>
1_11. fldProductSubGroupDescription  <VARCHAR>
1_12. fldMerch  <VARCHAR>
1_13. fldBrandName  <VARCHAR>
1_14. fldBaseProductDescription  <VARCHAR>
1_15. fldPC  <VARCHAR>
1_16. fldCL  <VARCHAR>
1_17. fldBrand  <VARCHAR>
1_18. fldAisle  <VARCHAR>
1_19. fldSectionLct  <VARCHAR>
1_20. fldShelf  <VARCHAR>
```

```
364. Mcvities jaffa cakes mini rolls 10 pack
365. Mcvities penguin mini rolls 10 pack
366. Cadburys banana mini rolls 6 pack
367. Cadbury dream cake bars 5pk
368. Mcvities jaffa cake bars 5 pack
369. Mr kipling chocolate chip cake bars 5 pack
370. Cadburys milk chocolate cake bars 5 pack
371. Cadburys caramel cake bars 5 pack
372. Cadburys 10 cake bar selection
373. MCVITIES MILKY WAY CAKE BARS 5 PACK
374. Mcvities galaxy cake bars 5 pack
375. Cadburys choc chunk muffin bar
376. Mcvities jaffa cake bars 10 pack
```

```
ch
lds
ff
t.
dvd
ml
t
hmk
cd
crl
p/m
trs
p
bd
ltr
ss
h/l
**
m
l
l'oreal
ckn
s
v
o/s
xl
s/s
finest*
tr**
pm
xxl
```

```
promo
occ
gby
schwartz
neutr
veg
sce
cont
hotshop
carlton
max
rimmel
nivea
cadburys
maybelline
teabags
excl
cond
wipes
uw
cami
jeandenim
apd
hartz
turq
issylace
mcvities
bio
bi
asst
```

```
tomatoes => tomato&(+)tomatoes&(+)tomt&
bread => brd
sliced => slcd(+)sld
white => wht
thick => thk
medium => med(+)m
extra => ex
bottle => btl
smoked => smkd(+)smk
bottles => btls
cheese => chse
```

```
Enquiry: white thick sliced bread
SQL Query: WHERE clause with AND operator
SELECT fldBaseProductDescription
FROM tblProduct
WHERE (fldBaseProductDescription LIKE '%white%' AND fldBaseProductDescription LIKE '%thick%' AND fldBaseProductDescription LIKE '%sliced%' AND fldBaseProductDescription LIKE '%bread%') OR (fldBaseProductDescription LIKE '%wht%' AND fldBaseProductDescription LIKE '%thk%' AND (fldBaseProductDescription LIKE '%slcd%' OR fldBaseProductDescription LIKE '%sld%') AND fldBaseProductDescription LIKE '%brd%')
```

Figure 3. Text data verification and mining

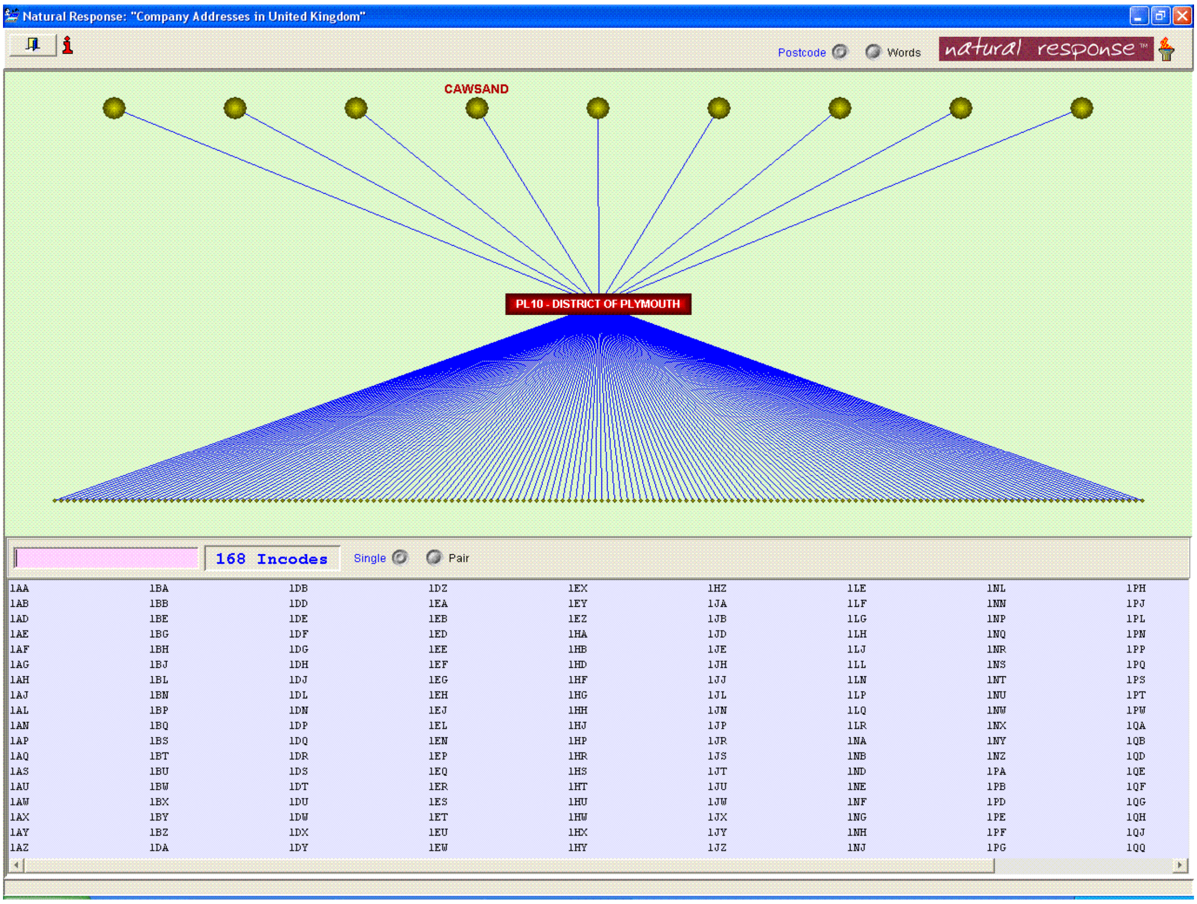
User's Enquiry Conversion. For an explanation it is better to consider some examples. Assume user's enquiry was "*White thick sliced bread*". Four appropriate PR have been allocated ("*white* ⇒ *wht* ", "*thick* ⇒ *thk* ", "*sliced* ⇒ *slcd* ⊕ *sld*", and "*bread* ⇒ *brd*"). Customer's data might contain any combination of wrong and correct words.

Theoretically for the considered example there are 16 possible combinations of data, namely: (1) "*White thick sliced bread*", (2) "*White thick sliced brd*", … , (16) "wht thk (slcd ⊕ sld) brd". Result of such conversion is shown on Figure 3.

Grammar Creation

Let us consider three aspects of grammar creation. The first one is regarding the ability of SR to better recognise (i.e. more reliable, with higher score) a sequence of words than single words e.g. $\mathcal{G}$ = *{white bread}* provides better recognition of VE "*white bread*" than $\mathcal{G}$ = *{white, bread}*. But for reliable recognition it is not enough to include in $\mathcal{G}$ just product names because the user is absolutely free to ask whatever he/she wants about, for instance, "W*hite thick sliced bread*", namely, "*white bread*", "*sliced bread*", "*thick sliced bread*" etc. On the one hand SSS (Sequential/Simultaneous Structure [5]) allows us to generate sensible sequences, on the other hand, the size of $\mathcal{G}$ is critical. A complex grammar usually causes a less accurate recognition because of the larger number of possible word sequences. The solution is to extract frequently used utterances from the users' VE. The important point is – any NR system, which provides VD, should be a self-learning system. Indeed, during the working day thousands and thousands of telephone calls to AD "Product" are taking place. "Sleeping time" is more than enough to update: (1) set of synonyms; (2) enquiry patterns and (3) grammar.

(white_sliced_bread) {white, sliced, bread, wht, slcd ⊕ sld, brd}.


Natural Response: "Company Addresses in United Kingdom"
Postcode
Words
natural response™
CAWSAND
PL10 - DISTRICT OF PLYMOUTH
168 Incodes
Single
Pair

1AA	1BA	1DB	1DZ	1EX	1HZ	1LE	1NL	1PH
1AB	1BB	1DD	1EA	1EY	1JA	1LF	1NN	1PJ
1AD	1BE	1DE	1EB	1EZ	1JB	1LG	1NP	1PL
1AE	1BG	1DF	1ED	1HA	1JD	1LH	1NQ	1PN
1AF	1BH	1DG	1EE	1HB	1JE	1LJ	1NR	1PP
1AG	1BJ	1DH	1EF	1HD	1JH	1LL	1NS	1PQ
1AH	1BL	1DJ	1EG	1HF	1JJ	1LN	1NT	1PS
1AJ	1BN	1DL	1EH	1HG	1JL	1LP	1NU	1PT
1AL	1BP	1DN	1EJ	1HH	1JN	1LQ	1NW	1PW
1AN	1BQ	1DP	1EL	1HJ	1JP	1LR	1NX	1QA
1AP	1BS	1DQ	1EN	1HP	1JR	1NA	1NY	1QB
1AQ	1BT	1DR	1EP	1HR	1JS	1NB	1NZ	1QD
1AS	1BU	1DS	1EQ	1HS	1JT	1ND	1PA	1QE
1AU	1BW	1DT	1ER	1HT	1JU	1NE	1PB	1QF
1AW	1BX	1DU	1ES	1HU	1JW	1NF	1PD	1QG
1AX	1BY	1DW	1ET	1HW	1JX	1NG	1PE	1QH
1AY	1BZ	1DX	1EU	1HX	1JY	1NH	1PF	1QJ
1AZ	1DA	1DY	1EW	1HY	1JZ	1NJ	1PG	1QQ

Figure 4. Towns and Incodes allocation

The second aspect is regarding the NR reaction. It goes without saying that VD must be in real time. Any delay of VD makes NR unacceptable. The obvious way of VE processing is: (1) recognise spoken utterance; (2) using PR add WW and UW; (3) convert VE+WW+UW to corresponding SQL query; (4) run SQL query; (5) synthesize reply; (6) provide text-to-speech conversion. We can avoid step (2) and include PR in grammar. Any line of grammar might contain two parts: *(spoken word or utterance) {result, or tag}*. The tag result is interpreted as a string and is delimited by braces "**{}**". All characters within the braces are considered as parts of the tag, including white space. The line of G for utterance *white sliced bread* will look as

The third aspect is devoted to dynamically created grammars. Assume user said his outcode *PL10*. Now instead of using the full size grammar (= 4,000 items) for Incodes it is better to allocate a set of Incodes regarding *PL10* (see Figure 4) and create a grammar just with 168 items.

Conclusion

VD and WD are *"the two sides of a coin"*, the features of which should complement each other. The main purpose of our report was to call attention of scientists to distinctive features of VD. As we know from our scientific experience it is always easier to find the best solution when you have an access to some practical results. That is why we included in our report real results but not ideal, or "dream" ones.

Bibliography

[1] G.A. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity to Process Information. *Psy.Rev.*,63,81-97,1956.

[2] Natural Response™. www.naturalresponse.net, 2002.

[3] T.M. Khalil, V. Lovitsky, "A Structure of Memory in Concept Formation", Proc. of the IEEE Conference on *Systems, Man and Cybernetics*, Boston, Mass,509-512,1973.

[4] J.Hoffmann, Das Aktive Gedachtnis. Psycologische Experimente und Theorien zur Menschlichen Gedachtnistatigkeit, *VEB Deutscher Vergal der Wissenschaften*, Berlin, 1982.

[5] В.А.Ловицкий, Классификация структур памяти, *Проблемы бионики*, Харьков,8,138-153,1972.

[6] WordNet®. http://wordnet.princeton.edu.

Authors' Information

David Burns – Natural Response (Natural Response is a trading style and trademark of RSVP Dialogue Limited), The i-zone, Bolton Institute, Dean Road, Bolton, Great Manchester, BL3 5AB, United Kingdom, e-mail: david@naturalresponse.net

Richard Fallon – Natural Response, The i-zone, Bolton Institute, Dean Road, Bolton, Great Manchester, BL3 5AB, United Kingdom, e-mail: richard@naturalresponse.net

Phil Lewis – Natural Response, The i-zone, Bolton Institute, Dean Road, Bolton, Great Manchester, BL3 5AB, United Kingdom, e-mail: phil@naturalresponse.net

Vladimir Lovitskii – Natural Response, The i-zone, Bolton Institute, Dean Road, Bolton, Great Manchester, BL3 5AB, United Kingdom, e-mail: vladimir@2ergo.com , vladimir@naturalresponse.net

Stuart Owen – Natural Response, The i-zone, Bolton Institute, Dean Road, Bolton, Great Manchester, BL3 5AB, United Kingdom, e-mail: stuart@naturalresponse.net