From Acts and Topics to Transactions and Dialogue Smoothness

Hans Dybkjær and Laila Dybkjær

Prolog Development Center A/S (PDC)
H. J. Holst Vej 3C-5C
2605 Brøndby, Denmark
dybkjaer@pdc.dk

Natural Interactive Systems Laboratory
University of Southern Denmark
Campusvej 55, 5230 Odense M, Denmark
laila@nis.sdu.dk

Abstract

Measuring transaction success and dialogue smoothness is extremely time consuming and costly when done manually and on many dialogues but is the only possibility today for spoken dialogue systems without a very clear success state. This paper investigates the possibility of automatic derivation of transaction success for task-oriented dialogues based on simple act-topic annotations.

1. Introduction

A key concern when bringing a spoken dialogue system (SDS) to the market is to ensure a high transaction success (TS) rate and a smooth dialogue. The TS rate measures the success of the SDS in providing users with the information they require and is an important quantitative parameter in ensuring user satisfaction. A smooth dialogue lets the user achieve his goals straightaway without any misunderstandings or other miscommunication and is one of the qualitative cost components of user satisfaction. If the dialogue is not smooth, users may still be dissatisfied even if they eventually succeed with their goals.

Measuring TS and dialogue smoothness is extremely time consuming and costly when done manually and on many dialogues. Manual annotation of TS and dialogue smoothness is the only possibility today for SDSs which do not have a very clear success state.

We propose a two-step method for computing transactions and to get an impression of dialogue smoothness. The *first* step is to annotate individual system and user utterances with a basic act-topic structure. We have done this manually so far but claim that the process is simple to automate by using a parser. The *second* step is to iteratively transform act-topic patterns into transaction segments.

We present formal patterns and discuss how expressive features are needed based on experiments with human-computer dialogue data. We conclude by outlining what is needed for full automation and further generalisation.

2. Background

Our interest in automatic markup of transaction success was aroused by a frequently asked questions (FAQ) SDS. This SDS was developed for FerieKonto - a Danish public institution that administers holiday allowance for many Danish employees - by Prolog Development Center (PDC) and NISLab. The system is able to provide various general information on holiday allowance and answer questions such as "Is Saturday considered a holiday" or "Can I transfer holiday to next year".

The FAQ system is an over-the-phone SDS implemented on the SpeechMania platform (now Scansoft) that includes the grammar and dialogue description language HDDL [Aust et al. 1995]. The system was put into production by the end of 2002 as the so far most advanced Danish commercial SDS.

The FAQ comprises 85 concepts and 233 prompts, including 66 holiday allowance rules, structured into 102 stories. A concept is a semantic abstraction that HDDL maps to concrete input grammar fragments. A story is a combination of several prompts into a larger structure. The dialogue model is represented in XML (about 2400 lines) and compiled into 12.000 lines of compact HDDL code, full of macro calls. In addition there are 2700 lines of grammar not included in the dialogue model.

A requirement in the contract for the FAQ SDS was the achievement of a certain minimum TS rate. We were faced with two problems here: The notion of transaction was not clearly defined, and we had no baseline for the measurement of transaction success [Paek 2001] in terms of e.g. recorded human-human dialogues.

Transaction success has for many years been used as a measure of the success of a SDS in providing users with the information they require, see e.g. [Danieli and Gerbino 1995]. The measurement of TS has often been made in terms of dialogue level task completion. This works for systems with a single, well-defined task and a clear goal state, such as simple train timetable information or flight reservation systems. However, in other cases, such as the FAQ system which contains many independent tasks, it is not possible to define one single goal state for a system. Each task may or may not be addressed during a dialogue and there is no particular order in which tasks are addressed. If a user addresses several tasks some of them may be correctly solved while others are not. In such cases it makes more sense to define transactions at sub-task level as also proposed by Brey et al. [2000] who address evaluation of a call centre automation system. An additional advantage of looking at transactions at sub-task level is that more detailed information can be obtained about which parts of the system function well and which ones seem to be problematic. Furthermore the metric of sub-task completion per time interval can be used on different versions of the system to measure efficiency.

Considering TS at sub-task level still does not define precisely what a transaction is. For instance, are repairs of miscommunication allowed as part of a transaction and are there any closeness restrictions, i.e. if the requested information is not achieved the first time but if the user later requests the same information again, do we then have one or two transactions, and does this depend on how far apart the two requests are?

Also, the mere annotation of a transaction simply by marking its start and end where the end may be either a success or a failure, does not provide much information about what went in between, i.e. about exchange patterns, potential problem patterns and dialogue smoothness.

In developing the FAQ system we approached these problems by creating an act-topic annotation scheme. We

Proceedings LREC'04, Lisbon, Portugal

then defined transactions in terms of (informal) patterns of act-topics, where a transaction may either end as a success or a failure. Using a coding tool tailored to the purpose we manually annotated about 225 dialogues from 3 test iterations, and 217 dialogues from the production system based on the annotation scheme [Dybkjær and Dybkjær 2002]. However, manual annotation is slow and the quality to some extent depends on the human annotator. Therefore we began to investigate the possibility of automating the act-topic and TS annotation process.

3. Towards Automatic Annotation of Act-Topics and Transactions

Brey et al. [2000] claim that sub-task success rate cannot be automatically annotated because it is a matter of interpretation. Success can only be measured by manual log file inspection of the attempt on the user's side to accomplish a given sub-task where an attempt may include multiple repair actions.

Walker et al. [2001] on the other hand suggest that a careful definition of transaction success, based on automatic analysis of events in a dialogue, such as acknowledgement of a booking, might serve as a substitute for hand-labelling of task completion. Using a dialogue act parser on a set of Communicator dialogues concerning flight reservation, hotel reservation and car rental, they are able to classify each system utterance according to speech act, sub-task, and conversational domain. Hastie et al. [2002] describe a continuation of this work and report on how the annotated system utterances are then used as a basis for automatic annotation of task-completion. Automation is based on looking for particular acts among the annotated utterances that serve as a kind of landmarks and indicate that a particular point in the dialogue has been reached e.g. that a flight itinerary has been accepted by the user.

The work by Hastie et al. [2002] shows that at least some kind of automatic TS annotation is possible They infer task completion from the tagging of specific system utterance states, but they disregard user utterances. This is a shortcoming of the approach because it means that in principle task completion need not be equal to task success. In the extreme case the user may never have been understood correctly. Moreover, the approach does not consider dialogue smoothness in much detail though start-over in terms of repeated requests for a trip is mentioned.

Based on our experience from the manual annotations we decided to explore and possibly justify the following three claims:

1. Act-topic annotation at utterance level can be automated. System utterances that are template based, can be tagged easily. User utterances may pose more problems, but the limitations imposed by the domain, even in a FAQ SDS, will make this feasible.
2. Transactions can be derived from act-topic patterns, and from these TS rates can be computed.
3. Act-topic patterns can contribute to the labelling of problem locations and to the metrics of smoothness.

Our approach to automatic annotation of act-topics and transactions has two major steps. *First*, some basic act-topic annotation must be added to all system and user utterances in a dialogue, cf. claim 1. For this paper, this annotation has been done manually. *Second*, basic acts are combined into composite acts and then further combined into transaction segments tagged with success or failure. The composite act-topic scheme and the transaction scheme both consist of a number of goals where a goal is met if its accompanying rules apply to the dialogue.

To enable the automatic application of a scheme to a dialogue we have implemented a program (actTopic.exe) in Visual Prolog 6.1. The following three sections describe our experiments and results in more detail.

4. Barebones Act-Topics

The simplest scheme only names the topics *T* in each utterance and distinguishes only 6 acts, cf. Figure 1. "Inform {T*}" indicates any topical information and thus is a very broad category encompassing most system as well as user utterances. "Other{T*}" is used about input which is neither task-related nor meta-communication. For instance the user exclamation "I'm talking to a computer!" would be categorised as an "other" speech act. "Pause{}" describes silence in input or output. "Hangup{}" is used when the user hangs up or when the system disconnects.

A *dialogue* is a sequence of turns of utterances, see Figure 1. Here italics denote non-terminals, '*' is zero-based repetition, '.' defines a value, *text* is the transcription text, and ':' and '{','}' are part of the syntax.

```
dialogue = turn* .
turn = utterance* .
utterance = who : act topics "text" .
who = .s | .u .
act  = .accept | .reject | .inform |
       .other | .pause | .hangup .
topics = {topic*} .
topic = T.name .
```

Figure 1. Formal structure of a dialogue.

A *pattern* is a list of utterances but may contain variables (prefix '_') for *who*, *act*, *topics*, and topic *name*. A pattern has a set of conditions. In the barebones case only *equality*, *member* and *non-equality* are included. {T_b} will match any topic list containing the act that T_b becomes bound to. Patterns may be described as a first-order unification based constraint logic.

The above basic acts are simple and can be added automatically and context-independently to utterances in a dialogue as a first step in our approach. System utterances can be annotated simply by adding the automatic annotation as part of the SDS output, i.e. each piece of output may be tagged at design-time and the annotation is then automatically added in the log-files at run-time. User utterances will have to be annotated automatically afterwards. This may be done by the SDS parser or a similar parser.

The act-topic annotation scheme used in the second step presumes that dialogues are annotated with the basic acts. ActTopic.exe applies the act-topic scheme rules, cf. Figure 2, to the step one annotated dialogues and transforms the basic acts into composite acts, cf. Figure 3.

```
rule request
   _y: .request {T_b}
      <-
      _x: .inform Ts_a
      _y: .inform {T_b}
   where
      T_b not-in Ts_a
      _x != _y
end rule
```

```
rule select1
   _y: .select Ts_a
      <-
      _x: .inform Ts_a
      _y: .accept {}
   where
      _x != _y
end rule
```

Figure 2. Act-topic scheme rules for composite acts.

```
.s: .inform {T.pay}
   "Payment in general"
.u: .accept {}
   "Yes"
```

```
.u: .select {T.pay}
   <- select1
   .s: .inform {T.pay}
   .u: .accept {}
```

Figure 3. Result (right column) of applying the select rule (Figure 2) to two basic acts (left column).

A problem in only identifying topics is that basically we can only distinguish between two composite acts, i.e. select and request, where select means continue with the same topic while request means change to another topic. For example, we cannot distinguish between request and repair which at this abstract level have the same pattern.

Figure 4 illustrates another problem in only looking at topics at a very overall level. In the left column we may consider the system's utterance as feedback. If the dialogue ends here it is a failure. In the right column we have a success. However, according to the act-topic annotation the two situations are identical. This means that we are not in a position to mark up transactions successfully unless we add more information.

```
.u: .inform {T.phone}
   "Your phone number?"
.s: .inform {T.phone}
   "Phone number"
```

```
.u: .inform {T.phone}
   "Your phone number?"
.s: .inform {T.phone}
   "Phone 48204910"
```

Figure 4. The two excerpts have the same basic annotation but different transaction states.

5. Distinguishing Name and Value

To allow for transaction annotation we started to distinguish between topic names and topic values. By a topic *name* we understand the mentioning of a topic, e.g. in terms of a user requesting information about a certain topic. By a topic *value* we understand details about a certain topic, e.g. the system informing about a topic name selected by the user.

The distinction between topic names and topic values enable us to write composite rules so that we can distinguish the two cases from Figure 4, cf. Figure 5.

```
.u: .inform {N.phone}
   "Your phone number?"
.s: .inform {N.phone}
   "Phone number"
```

```
.u: .inform {N.phone}
   "Your phone number?"
.s: .inform {V.phone}
   "Phone 48204910"
```

Figure 5. Distinction by topic name and value.

Figure 6 shows two different rules from the act-topic annotation scheme. The left-hand rule is applicable to the left column in Figure 5 and results in a select act. The right-hand rule is applicable to the right column in Figure 5 and results in a request act followed by an inform act.

```
rule select2
   _y: .select {N_b}
      <-
      _x: .inform {N_b}
      _y: .inform {N_b}
   where
      _x != _y
end rule
```

```
rule answer
   _y: .request {N_b}
   _x: .inform Vs_a
      <-
      _y: .inform {N_b}
      _x: .inform {V_b}
   where
      _x != _y
end rule
```

Figure 6. Rules applicable to the excerpts in Figure 5.

As a next sub-step we may apply a transaction annotation scheme to the dialogues. The left-hand rule in Figure 7 shows that the select act must be followed by an inform act on topic value in order to become a success. Thus in this case we don't have a full transaction yet and we don't know if it will become a success or a failure. The right-hand column shows a rule that matches the right-hand dialogue in Figure 5 after application of the right-hand rule in Figure 6. Thus in this case we have a TS.

```
rule success1
   _y: .success {N_b}
      <-
      _x: .select {N_b}
      _y: .inform Vs_a
   where
      _b in Vs_a
      _x != _y
end rule
```

```
rule success3
   _y: .success {N_b}
      <-
      _x: .request {N_b}
      _y: .inform Vs_a
   where
      _b in Vs_a
      _x != _y
end rule
```

Figure 7. Examples of transaction rules.

The distinction between topic names and topic values allows us to get much further regarding automatic annotation of transactions than was possible by only identifying topics in general.

6. Covering the FAQ

In order to completely cover act-topic annotation of dialogues with the FAQ system and the subsequent TS annotation we have added testing on certain topic names, i.e. `again`, `more`, and `closing` to the act-topic scheme described in Section 5. This enables us to capture e.g. requests for repetition and dialogue ends.

The current schemes contain 13 act-topic rules and 9 success/failure rules, averaging about 10 lines per rule. This was sufficient in a test run on 12 very different FAQ dialogues. Though the schemes remain to be tested on an independent and larger set of dialogues, they exhibit so far a surprising efficiency despite their simplicity.

7. Comparison to a Booking Task

An obvious question is to which extent our approach can be generalised to other task-domains. As a tiny experiment we considered the flight ticket reservation task and hand-annotated a representative human-computer dialogue from the Danish Dialogue System [Bernsen et al. 1998] with the step one basic speech acts. We then applied the act-topics and transaction schemes described in Sections 5 and 6 to the dialogue. Basically this worked but two issues had to be solved: sub-topics and composite tasks.

7.1. Sub-topics

The sub-topic problem relates to departure and destination cities. If the user only provides a city name in reply to e.g. a system question concerning departure city, we cannot without drawing on the context decide if the topic name of the user utterance should be to or from. We can only more abstractly decide that it is a place.

To be able to automatically relate the question and the reply in such situations we need to introduce the sub-topic relation `T1 < T2`. This is illustrated in Figure 8, assuming that the relation "`from < place`" has been declared in the act-topic scheme.

```
.s: .inform
   {N.travel, N.from}
  "Where does the travel
   start?"
.u: .inform {V.place}
  "Copenhagen"
```

```
rule selectSub1
  _y: .select {N_a}
  <-
    _x: .inform {N_a}
    _y: .inform {T_b}
  where
    _a < _b
    _x != _y
end rule
```

Figure 8. `V.place` is a proper response to `N.from` (left). The select pattern applies if sub-concept is used instead of equality (right), compare `select2` in Figure 6.

7.2. Composite tasks

Regarding the FAQ dialogues we have only looked at transactions for tasks which are not in a hierarchical relation to a super-task. However, if we move e.g. to flight ticket reservation this task may contain a number of sub-tasks or micro-transactions such as departure, destination, day and hour. Each sub-task may be a success or a failure but we may also want to bind all these sub-tasks together in a super-task which covers a ticket reservation and which may be a success or a failure depending on the sub-tasks. It is possible to add a rule which subsequent to transaction annotation at sub-task level and depending on whether all required sub-tasks have a success, tags the entire dialogue as a success or a failure.

8. Conclusion and Next Steps

We have presented (parts of) an automatic transaction annotation process based on transformations of dialogue act-topic patterns. We have constructed a program that annotates composite act-topics and transactions on the basis of basic act-topic annotation of a dialogue and annotation scheme rules. We developed the act-topics and transaction schemes for a FAQ task domain for which they seem to work quite well. We further made a tiny experiment and tried them on a reservation dialogue. Needed extensions of the schemes are described.

To obtain a fully automatic annotation process, we must:

- Parse dialogues to produce basic act-topic annotations.
- Combine the parser and pattern transformer into an automatic batch system.
- Test the system (parser and pattern transformer) on a larger number of FAQ dialogues.
- Test the system further on other kinds of dialogues (reservation, travel information, etc.).

8.1. Acts or topics

The act-topic scheme described concentrates on topic information. Note how the name/value distinction and even more the testing for specific topics like `again`, moves the topic interpretation into the act dimension. Roughly, topics are seen as belonging to the utterance level whereas acts, via the patterns, relate to the discourse level.

It is the uniform simplicity of topics that makes us claim that it is sufficient to parse each utterance out-of-context. We may pragmatically exploit the knowledge of what the system utterances are intended to do in making the parsing rules, at the expense of generality.

We note that our purpose is to provide an approximately correct transaction success rate in a concrete context. For this, the current abstract description seems sufficient. A detailed and rich structure describing the conversation *per se* is not needed.

8.2. Other features

Our patterns allow us to match explicitly on `s` and `u`. However, all rules are stated symmetrically using `x` and `y`, despite that the FAQ is inherently asymmetric in system and user. Some rules may tacitly exploit this asymmetry and fail to work in dialogues with symmetric speakers. However, this kind of dialogues fall beyond our scope.

We have not considered negation. Locally, we may detect topic negation and use the current pattern formalism.

8.3. Smoothness

Some of the rules in our act-topic scheme produce acts which point to problems in the dialogue. Examples are repair acts and repeated requests for the same information, cf. also the start-over problem mentioned by [Hastie et al. 2002]. Of course a transaction failure also contributes negatively to smoothness. However, for the moment we are not sure to which extent we should count e.g. requests for repetition and system clarification questions as negative contributors to smoothness. Once we know which acts and transactions to look for, it is easy to count how many of them there are. But a next step would be a further investigation of which they are and how much they contribute.

References

[Aust et al. 1995] Harald Aust, Martin Oerder, F. Seide and V. Stenbiss: The Philips automatic train timetable information system. *Speech Communication* 17, 1995, 249-262.

[Bernsen et al. 1998] Niels Ole Bernsen, Hans Dybkjær, and Laila Dybkjær: *Designing interactive speech systems. From first ideas to user testing*. Springer Verlag 1998.

[Brey et al. 2000] Thomas Brey, Gerhard Hanrieder, Paul Heisterkamp, Ludwig Hitzenberger, and Peter Regel-Brietzmann: Issues in the evaluation of spoken dialogue systems - Experience from the ACCeSS project. *Proceedings of LREC 2000*, 731-734.

[Danieli and Gerbino 1995] Morena Danieli and Elisabetta Gerbino: Metrics for evaluating dialogue strategies in a spoken language system. *Proceedings of the AAAI Spring Symposium Series* (Empirical Methods in Discourse Interpretation and Generation), 1995.

[Dybkjær and Dybkjær 2002] Hans Dybkjær and Laila Dybkjær: Measuring transaction success in spoken dialogue information systems. *Proceedings of Nordtalk Symposium on Relations between Utterances*, Copenhagen, December 2002, 110-131.

[Hastie et al. 2002] Helen Wright Hastie, Rashmi Prasad, Marilyn Walker: Automatic evaluation: Using a DATE dialogue act tagger for user satisfaction and task completion prediction. *Proceedings of LREC2002*, 641-648.

[Paek 2001] Tim Paek: Empirical methods for evaluating dialog systems. *Proceedings of the 2nd SIGdial workshop on discourse and dialogue*, Aalborg, Denmark, 1-2 September 2001, 100-107.

[Walker et al. 2001] Marilyn Walker, Rebecca Passonneau, and Julie Boland: Quantitative and qualitative evaluation of Darpa Communicator spoken dialogue systems. *Proceedings of ACL 2001*, Toulouse, France, 2001, 515-522.