Follow-up utterances in QA dialogue

Boris van Schooten — Rieks op den Akker

Faculty EWI, Universiteit Twente, Enschede, Netherlands
{schooten,infrieks}@cs.utwente.nl

ABSTRACT. *The processing of user follow-up utterances by a QA system is a topic which is still in its infant stages, but enjoys growing interest in the QA community. In this paper, we discuss the broader issues related to handling follow-up utterances in a real-life "information kiosk" setting. With help of a corpus of follow-up utterances, we lay out a general framework for QA dialogue, and propose a general-coverage classification of real user reactions, based on their function in the dialogue, and their relation to the context of previous utterances. We discuss a dialogue manager we have developed that handles the identified types of follow-up utterances. It uses existing NLP and IR techniques to handle follow-up questions, and generates appropriate dialogue reactions to handle non-follow-up questions. Our main conclusion is that a large part of follow-up utterances can be handled by a set of simple techniques, even though we found it is not easy to determine the exact context in which follow-up questions are to be understood.*

RÉSUMÉ. *Le traitement des énoncés suivantes d'un utilisateur par un système de question/réponse est une thème qui est à ses premières parties, quoique l'intérêt de la communauté de QA s'accroît. En cet article, nous discutons les sujets larges qui sont liés au traitement des énoncés suivantes dans un arrangement réel d'un «guichet d'information». Nous présentons un cadre général pour le dialogue de question/réponse, qui nous avons développé avec l'aide d'un corpus des énoncés suivantes. Nous proposons une classification général des réactions d'un vrai utilisateur, basée sur leur fonction dans le dialogue, et sur leur relation avec les énoncés précédentes. Nous discutons un gestionnaire de dialogue développé par nous qui traite les types identifiés des énoncés suivantes. Il emploie des techniques existants de NLP et de IR pour manipuler des questions suivantes, et il produit des réactions propres de dialogue aux autres énoncés suivantes. Notre conclusion principale est qu'une grande partie des énoncés suivantes peut être traité par quelques techniques simples, quoique nous avons trouvé il n'est pas facile de déterminer le contexte exact dans lequel on doit comprendre les questions suivantes.*

KEYWORDS: *question answering, dialogue management, follow-up questions, sentence rewriting*

MOTS-CLÉS : *question/réponse, gestion de dialogue, questions suivantes, réécriture de phrases*

1. Introduction

In the research described here, we aim at a QA dialogue system that is useable in real life "information kiosk" applications for non-expert users. In this paper, we argue that we have yet to bridge a gap between current QA dialogue systems, and actual user behaviour we can expect in real QA dialogues. This paper is a first step towards bridging this gap.

The research described here is part of the IMIX project, which brings together research on Dutch language technology: QA, dialogue management, speech recognition and generation, and information presentation. As part of the project, we develop a QA dialogue system for the medical domain (Akker *et al.*, 2005), called the IMIX demonstrator. The demonstrator contains three parallel QA engines from three different QA projects: Rolaquad (Lendvai, 2005), QADR (Bouma *et al.*, 2006), and Factmine (Tjong Kim Sang *et al.*, 2005). The dialogue manager, which will be discussed here, is developed as a separate module.

IMIX focuses on a closed domain, the medical domain. However, many of the techniques used are also applicable to open-domain QA. This is especially true of the dialogue management module. While the system is Dutch-language, most of the findings of the research described here are very likely to be extensible to other languages. We will compare our findings to research on different languages where appropriate.

Existing dialogue management for QA follows two basic strategies: follow-up questions (enabling the user to pose questions which are to be understood in the context of previous questions or answers) and system clarification questions (having the system ask the user for more information to improve information retrieval). This paper concentrates on follow-up questions (fuq). It describes a feasibility study of handling fuq in a real-life setting. In this study we consider the more general case of user follow-up utterances (fuu). Our first finding is that there are a number of distinct classes of fuu, and there is a gradual continuum between types of "classical" fuq and the types of fuu that "real" users utter. Our second finding is that these fuu can already be dealt with effectively by means of relatively simple, domain-independent, and knowledge-poor techniques. To illustrate this, we propose and discuss a nearly domain independent dialogue manager that handles fuu.

The dialogue setting we look at has the following properties:

– The user is not trained in using QA, and is not very knowledgeable in the domain.

– The user may pose any kind of domain questions, which may be other questions than factoid questions.

– The user will, and is in fact invited to, utter free-form follow-up utterances, and not just follow-up questions.

We will describe the broader context of our research by comparing it with the most common QA framework: the Trec/Qac/Clef framework. We depart from the classical Trec/Qac/Clef context in several ways:

– **Encyclopedic questions.** The trade-off between question type coverage and answering performance is an important issue for QA systems. In a perfectly modular approach to dialogue management, we could theoretically abstract away from specific question types, since we can delegate all technical details related to QA to the QA module. To an significant extent, this holds for the dialogue manager that we propose. In the QA contests, the focus on "factoid questions" has been a successful trade-off which enabled QA systems to be unambiguously evaluated and to evolve to a satisfactory level of performance. The QA contests continue to extend their coverage to more difficult types of questions, such as complex temporal (clef), list, and relationship questions. However, the coverage still makes a lot of artificial simplifications, and/or performance of current QA systems on these is still low (Herrera *et al.*, 2005; Nyberg *et al.*, 2005). The IMIX project has made particular choices regarding the nature and difficulty level of questions. We try to provide a coverage that is enough for real users of our target group. We call the types of questions we cover "encyclopedic questions". These are basically questions which can be answered by selecting an appropriate piece of text from a sufficiently large encyclopedia. Our system typically answers with a piece of text about 1-3 sentences long. Our system is not made to support the more difficult question types. Nevertheless, correctness of answers is less well-defined and harder to measure than with factoid questions. The length and detail level of a correct answer depends on the information need of the user. This need can not be precisely known beforehand. Enabling the user to give feedback on his/her information need is one of the purposes of our dialogue system.

– **A follow-up question can be a reaction to an answer.** While Trec and Qac attempted to include dialogue in their repertoire, the dialogue track in Qac and the former dialogue track in Trec are limited, and suffer from some methodological problems, which was the reason why the Trec dialogue track was dropped (Voorhees, 2001). They have a limited view of what a fuq is. All fuqs in the Trec/Qac tracks assume that each question always has exactly one answer, consistent with the "factoid" paradigm. The dialogues then assume that the user never needs to react to these answers, but follows a planned path. In reality, answers may have varying degrees of correctness or completeness, and users will respond to this with a continuum from pre-planned fuq to fuq concerning details of the answer, to utterances that indicate the user is unhappy with the answer.

– **We consider reactions to wrong or unclear answers.** Many answers will be plain wrong, and many will be only partial or unclear answers. Such bad answers should not cause the dialogue to disintegrate. The dialogue manager should at the least know something about users' reactions when they are confronted with wrong or unclear answers.

– **Not all meaningful follow-up utterances are regular follow-up questions.** Most QA research does not account for user reactions other than domain questions that are readily processable. In real-life dialogues, we see other phenomena: utterances that indicate uncertainty about the correctness of the answer, negative feedback, and acknowledgements.

We try to design our dialogue manager to be as generic as possible. In particular, we chose the dialogue manager to:

– **be domain knowledge poor.** This means that the dialogue manager does not require any domain knowledge, or only broad-coverage domain knowledge. This would mean it's generally applicable to any domain, and results found are more generalisable to other domains. Introducing either hand-coded or data-driven domain knowledge would introduce various issues related to the specific domain and method used. This does not mean that we will not consider incorporating semantic knowledge where possible and available, just that we prefer the cases where it isn't necessary. We shall give an account of the usage of semantic knowledge where appropriate.

– **depend as little as possible on a specific QA architecture.** Being independent of a QA strategy or architecture has the advantage that the dialogue manager's principles can be extended to both different QA implementations and different types of questions. In IMIX we have a nice starting point: in the demonstrator system, all three QA engines have equal input and output protocols. We have the ability to use all three of them in a dialogue, despite their differences. If we implement dialogue functionality that is compatible with all three engines, we are already a first step forward towards a QA-independent dialogue manager.

1.1. *A generic QA interface*

If we want to establish insights on QA dialogue which are independent of a specific QA architecture, we need to say something about what we view as a "generic" QA system.

Typical QAs have a separate information retrieval (IR) stage, but there are many differences between individual systems. In particular, the internal representation of the IR query varies. Typical QAs translate a question into an IR query by classifying it into a specific question type with appropriate arguments, others build a more complex semantic frame from the question (such as SmartWeb (Reithinger *et al.*, 2005) and Ritel (Galibert *et al.*, 2005)). The IR itself may be done in different ways, such as using semantic document tags resulting from a pre-analysis phase (such as IMIX's Factmine and Rolaquad), or by matching syntactic structures of question and answer (such as IMIX's QADR). Because of all these variations, we shall assume that a dialogue manager will only be able to give relatively simple and general hints to the QA back-end, such as pointers to the questions that should be considered context for the current question. The QA system is then responsible for the details of implementing this dialogue context into its QA query. We shall go into more detail on this issue in section 3.

It would be nice if QAs give a confidence score with each answer that enables a dialogue manager to make proper decisions. While some QAs only return one answer, many QAs give a list of confidence-ranked answers. However, the confidences are typically not comparable between systems, and the values are not always useable as

absolute measures of confidence. Since implementing universally normalised confidences properly is not trivial, requiring that QAs will return reliable confidence figures will be detrimental for a dialogue manager's universality. We will assume that the top answer returned by the QA is most likely to be the best one, and we will not consider confidence values in this paper.

A common factor that we shall assume is that all answers are (slightly modified) text fragments, taken from a document out of a set of documents. The documents are typically a few paragraphs of text on a specific subject. In fact, the IMIX database documents are mostly sections from medical encyclopedias.

1.2. *Overview of paper*

The paper is subdivided as follows. In section 2, we will propose a fuu classification, based on our "second question" corpus. In section 3, we will propose a further classification of the subclass of fuq found in our corpus. We will then analyse our classification scheme with help of annotator agreement analysis in section 4. In section 5, we describe and evaluate our dialogue engine. We conclude with section 6.

2. Classification of follow-up utterances

We argue that a real-life QA dialogue system should at least be able to distinguish between common types of fuu. To find out what kinds of fuu were common for naive users, we collected a corpus. We used a special method for collecting this corpus, which is low-cost and specifically tailored for fuu in QA systems. Instead of having an actual dialogue with the user (using either a dialogue system or a Wizard of Oz setup), we have the user react to a set of canned question/answer pairs. The first user turn consists not of posing a free-form question, but of selecting a question out of an appropriately large set of interesting questions. The second turn of the user consists of posing a fuu to the answer then presented. The dialogue simply ends when the user posed his/her fuu, and there is no need for the system to reply to the fuu, hence there is no dependency of the corpus on any dialogue strategies used. An obvious weakness of the method is that the dialogues are only two turns long. However, we found that such a "second dialogue turn" corpus can be rapidly collected, and contains many or most of the most important phenomena that a first version of a dialogue system will need to support. Our first conclusion is that this is a very good low-cost corpus collection method for bootstrapping a QA dialogue system.

We first created a collection of 120 hand-picked questions with selected answers. The collection was chosen so as to cover a variety of different question types and answer types, and is also being used to evaluate the IMIX QA engines. Answer size ranges from a single phrase to a paragraph of text. The answers had a proportion of fully or mostly correct answers (93 questions, the answers were retrieved manually), a proportion of wrong answers (20 questions, the answers are real output from the

QA system), and a proportion of "no answer found" (7 questions). The users participated in the experiment through a Web interface. First, they had to select at least 12 questions which they found particularly interesting. Then the answers were displayed. For each question-answer pair, they had to enter a fuu that they thought would help further serve their information need, imagining they were in a real human-computer dialogue. We asked about 100 users to participate, which were mainly people from the computer science department and people working in a medical environment. We collected 575 fuus from 40 users. The questions chosen by the users were reasonably evenly distributed. Almost all questions were chosen between 1 and 10 times by users; there was no question that was not chosen by any user.

Examining the corpus, we soon found that the fuus could meaningfully be classified into a number of distinct classes. We annotated the corpus with these classes. To evaluate the validity of the classification, we also performed an inter-annotator agreement analysis, which will be described in section 4.

We found three main classes of fuu (See figure 1):

– **follow-up question (56%)**. We consider all domain questions that should be understood in the context of previous utterances, and which can meaningfully be interpreted literally, to be fuq. They illustrate that the user acknowledged the answer at least partially, and indicate a further user information need. Some of the fuq in our corpus contained cue words indicating their "follow-up" nature, but there were none with politeness forms or indirect forms. A significant part of these were effectively self-contained, even though they were clearly on the same topic (25% of all fuq).

– **negative feedback (28%)**. This includes negative feedback questions and statements, questions indicating uncertainty about correctness of the answer, and reformulations indicating the user was not happy with the answer. We found several distinct types of these:

1) **negative questions and statements 20%.** There seemed to be two main forms of these: repetition of the original question, with or without negative cue phrases (with no serious attempts at reformulations made); and a negative remark, usually simple but sometimes containing corrective information. In general, it appeared that there was relatively little useful information to be obtained by further analysing the negative feedback utterances. In some cases, a negative question and statement were combined in a single utterance.

q:	wat zijn hartkloppingen?	*What are heart palpitations?*
a:	De patiënt neemt dit waar als hartkloppingen.	*The patient experiences this as heart palpitations*
fuu:	Maar wat zijn hartkloppingen dan?	*But what are heart palpitations?*

Repetition example

2) **verify questions 3.6%.** Questions that indicate that the user is not sure about the meaningfulness or correctness of the answer.

q: Hoe merk je dat je hoge bloeddruk hebt?
       What do you notice when you have high blood pressure?
a: Hoge bloeddruk (hypertensie) is meestal een aandoening die over het algemeen geen symptomen veroorzaakt.
       High blood pressure (hypertension) is usually an affliction that generally does not cause any symptoms
fuu: Dus je merkt niet dat je een hoge bloeddruk hebt?
       So you don't notice anything when you have high blood pressure?
                              Verify question example

3) **reformulations (4.4%).** These usually occurred when the system gave a "no answer" response. They are generally self-contained questions without any special linguistic cues.

q: Komt RSI in Nederland vaker voor dan in de rest van Europa?
       Does RSI occur more often in the Netherland than in the rest of Europe?
a: RSI komt niet alleen bij beeldschermwerkers voor maar ook in de industrie en bouwsector.
       RSI does not only occur among screen workers but also in industry and construction.
fuu: hoe vaak komt RSI voor in nederland vergeleken met de rest van europa
       how often does RSI occur in the nederlands as compared to the rest of europe?
                              Reformulation example

– **acknowledgements (13%).**

Almost all acknowledgements consisted of a one- or two-word acknowledge phrase, such as "ok" or "thanks".

For the rest of the article, we name the above classes resp. **fuq**, **negative**, **verify-question**, **reformulation**, and **acknowledge**. Everything else was labeled as **other**; this covered only 2.8% of the utterances. About half of these could be classified as "meta" requests, such as asking for a literature source, or requests concerning search strategy or answer form.

So, we found that 44% of the utterances are not fuq. Recognising and dealing with these classes of fuu will already improve the system significantly. Even just reacting with appropriate prompts will help. In some cases, showing (more of) the document where the answer came from is a meaningful reaction. More sophisticated techniques can be imagined, involving system clarification questions for example.

Now, let's look at the distributions for the different types of answer correct/incorrect/no-answer (see figure 1). As we might expect, correct answers are replied mostly by fuqs, and incorrect answers by negative feedback and verify-questions, although a significant minority were fuqs. Almost half of the no-answers were spontaneously reacted to by reformulations, though the users were not prompted to do so. A quarter were acknowledgements, indicating the users accepted the absence of an answer. The "other" category was significant here. In fact, we found that almost all "other" utterances amounted to explicit requests to search again. A dialogue system could easily react to this by an appropriate reformulation prompt.
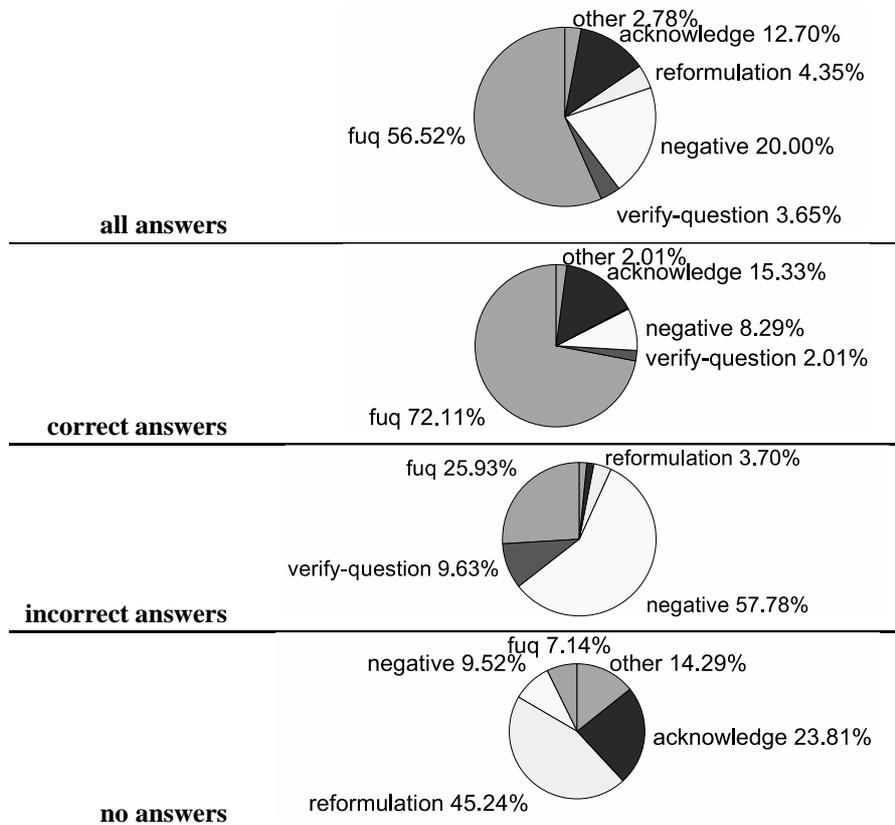
Figure 1. *Types of fuu in corpus. fuq = followup question. self-contained = followup question that can be understood without context. negative = negative feedback.*

The fuu class seems to be an indication of whether an answer is satisfactory and/or correct. It is interesting to find out to what extent we can discover the quality of the answer by just looking at the fuu class. Significant is that no strong conclusions can be drawn when the user poses a fuq. But if we look at the other fuu classes' potential of classifying between correct and incorrect answers, we can identify the following patterns:

– acknowledge almost always means correct.

– verify-question almost always means incorrect.

– reformulation almost always means incorrect.

– negative usually means incorrect. There is a small but significant percentage of negative feedback to answers labeled as correct. A look at these answers indicated that the quality of these was less than that of most answers labeled as correct. This appears to be in part due to the fact that the answers were limited to selected text fragments

from the corpus only. In these cases, a negative reaction is understandable. In fact, the user reactions we found can be used to reconsider the correctness of these answers in some cases.

These simple rules would allow us to determine answer correctness to some degree. In particular, we can predict a large part of the incorrect answers:

actual↓ *predicted*→	correct	incorrect	unknown
correct	15.3%	10.3%	74.4%
incorrect	1.2%	71.8%	27.0%

3. Handling follow-up questions

In this section, we will take a closer look at the subclass of fuq. As we shall see, different kinds of fuq can be distinguished that require different kinds of handling by a QA dialogue system. If we look at current QA dialogue research, we can distinguish two basic approaches to handling fuqs:

1) **rewriting a follow-up question to a self-contained question.** The greatest advantage of rewriting is its modularity: the QA can in principle remain a black box, even though determining the effectiveness of your sentence rewriting may require knowing something about the workings of the QA engine you are using. A second advantage is that a successfully rewritten question ensures that our interpretation of the fuq is correct and complete. Moreover, this correctness and completeness can be readily evaluated by a human annotator by simply judging whether the question is self-contained and answerable.

In general, sentence rewriting is tricky business. It turns out to be hard to get correct results. It is also hard to verify how good the results really are. A sentence can be rewritten in different ways, and we have to decide somehow which is the "best" one. For QA, one of the issues with rewriting is ensuring that the question remains comprehensible to the QA. A rewritten question will easily become complex, and the performance impact on the QA will be unclear. We can write down several competing criteria for the suitability of a rewritten question:

a) all appropriate search terms occur in the rewritten sentence

b) sentence is syntactically and semantically correct

c) sentence is as simple as possible

d) sentence is answerable by a human "QA"

e) the QA gives the correct answer

The criteria that we emphasise here are (b), (c), and (d). We believe (a) is insufficient given the number of non-search-term approaches, and is contained in (d). While empirical purists may consider (e) to be the "ultimate proof" of suitability, in reality it is not without problems, as it contains noise due to introduction of additional variables. In particular, the result depends on both the quality of the particular QA and the

document database used. The criteria (b)-(d) have the additional advantage that they can be evaluated readily by a human annotator.

Basic forms of rewriting include replacing an anaphor with a description of its referent, and adding missing phrases to elliptical or otherwise incomplete sentences. Such rewriting satisfies criteria (a)-(d). For example, the (Japanese) Rits-QA system (Fukumoto *et al.*, 2004; Fukumoto, 2006) uses two kinds of ellipsis expansion, and anaphor resolution. Their scheme managed to rewrite 37% of the fuqs in their corpus correctly.

2) **combining the fuq's IR query with that of previous utterances.** We will call this the *IR context* approach. This approach requires us to "break open" the QA system. The advantage is that we can take shortcuts, which may enable us to avoid having to fully interpret a fuq when not strictly necessary. In some cases however, rewriting and IR context amount to much the same thing, and require much the same techniques.

As an example of this approach, consider the (Japanese) Nara Institute system (Inui *et al.*, 2003). It handles fuq using IR context, based on its "question type/keyword" based IR:

a) Analyse the question, obtaining keywords, question type, and answer class. Obtain question type and answer class from dialogue history if missing.

b) Add keywords from dialogue history (from both system and user utterances).

c) Remove keywords with low weights, or with the same semantic class as answer class. For each semantic class, keep only the keyword with the highest weight.

d) If no answer is found, relax the current request until an answer is found.

Note that step (b)-(c) are similar to a regular salience-based linguistic reference resolution scheme, although they avoid having to resolve specific referents. The (English) KAIST system (Oh *et al.*, 2001) uses an approach similar to step (b)-(c), except that their reference resolution does resolve to a specific referent. Similar schemes are found in the (French) Ritel (Galibert *et al.*, 2005) and (German) SmartWeb (Reithinger *et al.*, 2005) system, both of which use merging of semantic frame representations. Ritel also uses request relaxation.

Simpler variants of the approach also proved quite fruitful. In particular, just searching within the top n documents retrieved by a previous question seems to be a successful strategy, as pointed out by De Boni (De Boni *et al.*, 2004). De Boni found that about 50% of fuq could be answered using this strategy (given a perfect enough QA). Such a facility would even enable users to use a strategy of incremental refinement, that is, using multiple simple questions to assemble a query throughout multiple utterances. Implementation-wise, this is similar to adding the search terms from the previous question to the IR query. So, this method will be easy to implement for any keyword-based IR, or any IR which returns multiple documents.

An even simpler method can be imagined, namely, searching only the document where the previous answer came from. This has some significant advantages in terms

of simplicity. In particular it's a method that will work for *any* QA engine that works by selecting text fragments from documents.

IR context methods may be trigged by a dialogue manager by simply indicating whether context from specific previous utterances needs to be used or not. De Boni used an even simpler trigger, namely a single flag indicating whether context should be used or not (indicating whether the current question is resp. a fuq or a new question).

3.1. *Our approach*

We will consider a multi-strategy approach, in which the dialogue manager gives simple hints to the underlying QA system, and the QA system uses the hints as appropriate. In particular:

1) a flag indicating whether this question requires context or not (this is the method used by De Boni)

2) the rewritten question, if it could be rewritten

3) a reference to the previous questions or answers that this question appears to refer to, if these could be determined

In order to get an idea how successful such a strategy would be, we looked at our corpus again. We split the fuq portion (56%) into classes, based on how they refer to the dialogue context. In particular, to find out if fuqs could be rewritten, we attempted to rewrite each fuq manually into a self-contained question satisfying rewriting criteria (b)-(d). We also identified several special subclasses of rewritable fuqs, based on the existence of machine-ready transformations that can be used to rewrite them. The transformations we identify are the following:

– **anaphor**: fuqs with anaphors which refer to NP antecedents,

– **elliptic**: elliptic fuqs (fuqs without verb) which could be expanded by including constituents from a previous sentence,

– **other-pp**: fuqs which could be expanded by attaching a PP at the end, consisting of a prep and an NP from a previous utterance, or which is a PP from a previous utterance.

The rewritable questions that did not fall into these categories we labeled as **referencing-other**. Our estimation is that most of these will be difficult to rewrite without rich knowledge. Some fuqs did not require rewriting, even though almost all of them are clearly within the context of the previous question or answer in terms of information need. We labeled these as **self-contained**.

One more surprising class emerged: namely, a significant number of fuq turned out to be really demands of the user to show a missing part of the text that the text fragment (implicitly or explicitly) referred to. These could not be rewritten except by actually quoting (parts of) the answer literally. We label these as **missing-referent**. An example:

q:	Waar duiden koude voeten op?	*What do cold feet indicate?*
a:	Hierdoor kan een verhoogde bloeddruk ontstaan of een vernauwing van de bloedvaten in het been.	*This can cause heightened blood pressure or a constriction of the blood vessels in the leg.*
fuu:	waardoor?	*What can?*

Missing referent example.

Figure 2 shows the breakup of fuq into the different classes.
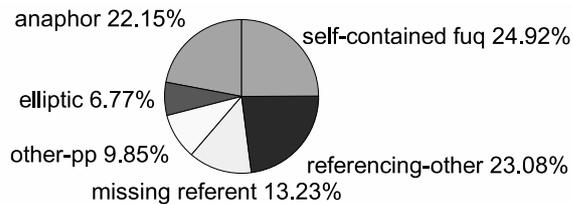

Figure 2. *Types of fuq in corpus. anaphor = question contains anaphor referring to NP, elliptic = question is elliptic (no verb) and can be expanded by adding constituents from a previous sentence, other-pp = can be rewritten by adding PP with NP from previous sentence, missing-referent = fuqs that requested for something missing in the text fragment, referencing-other = all fuq that could be rewritten, but not with any machine-ready technique, self-contained = fuq which can be understood without context*

What does this mean for our approach? Let's assume that the 13.2% of "missing referent" questions may be answered by just displaying more of the document the answer came from. We can say that, of the 61.9% of fuq that are not already self-contained nor of type missing-referent, some 62.7% are potentially rewritable using relatively basic techniques (that is, they belong to the classes anaphor, elliptic, and other-pp). This is an *upper bound* for the rewriting approach, assuming that we can correctly classify and rewrite the user utterances.

The remaining ones may alternatively be resolved using the IR context approach. How many of these will be resolvable in this way cannot easily be determined, and depends on the inner workings of the IR engine and database. In contrast to the rewriting approach, we cannot say with certainty if a certain IR context operation is "correct" and effective, given a specific QA engine.

We will look closer at "missing referent" questions and the IR context approach in section 5.

4. Annotator agreement analysis

Combining all the fuu and fuq classes, we arrive at a total of 11 classes. We have 5 non-fuq classes, **negative**, **verify-question**, **reformulation**, **acknowledge**, and **other**,

and 6 fuq classes, **anaphor**, **elliptic**, **other-pp**, **referencing-other**, **self-contained**, and **missing-referent**. To see if these classes are meaningfully distinguishable, we conducted an inter-annotator agreement analysis. This amounts to having multiple annotators annotate the same corpus using the same annotation manual, and seeing if they agree on the annotations chosen. This is arguably a required procedure for validating any classification scheme that is dependent on human annotation, and we can't help but notice that many such classification schemes fail to produce inter-annotator agreement results.

With respect to inter-annotator agreement, this classification has some non-standard properties. In particular, in some cases multiple interpretations are possible which are equally valid. For example, questions may be rewritten in different ways which are equally satisfactory. Also, disagreement about some pairs of classes may just be an indication that similar disagreements are to be expected in human-human dialogue (in particular, literal interpretation versus reacting to indirect intent).

We could solve such potential problems by preventively increasing the specification of the annotation scheme (and also its artificiality and difficulty level). We chose not to do so however, and instead decided to look at what patterns would emerge without such an overspecification, when the corpus is annotated by multiple annotators.

We instructed a number of annotators with a 1.5-hour interactive lecture and a 6-page annotation manual. Next to definitions of the classes with examples, the manual contained a decision tree (depicted in figure 3). The prescribed strategy was to first determine if the utterance was a fuq or one of the other classes, and, in case it is a fuq, to try to rewrite the question first, before deciding on the fuq subclass. The corpus was split into three portions of about 192 questions each, and each portion was annotated by a different annotator. We found the annotation takes an average of about 1 minute per fuu. So, it's a relatively slow annotation task, mainly due to the rewriting task. With 23 annotators in total, we obtained 7-8 annotations for each fuu.

Inter-annotator agreement is usually measured using a Kappa formula, which determines how the agreement relates to agreement by chance. A value of 0 means that the agreement is no better than agreement by chance, while 1 means perfect agreement. We compared the three most common Kappa formulas, Cohen's Kappa (averaging the Kappas over all pairs of annotators) (Cohen, 1960), Krippendorff's Alpha (Krippendorff, 1980) based on the coincidence matrices of pairs of annotators (again averaging over all pairs of annotators), and Alpha based on a single coincidence matrix of all annotators. They gave near identical results, with figures of about 0.62. A kappa of 0.7-0.8 is usually recommended for adequate reproducibility, which means we scored below the mark, although not prohibitively so. A global look at the annotations showed that there was significant noise present (in the sense that many annotations clearly violated the annotation rules), but that noise did not seem to be the main reason for the low Kappa score. Also, some interesting alternative rewritings were found by the annotators. In most cases though, annotators agreed on the rewritten sentence, except for the free-form "referencing-other" class. In some cases, annotators forgot to include all context in the rewritten question.
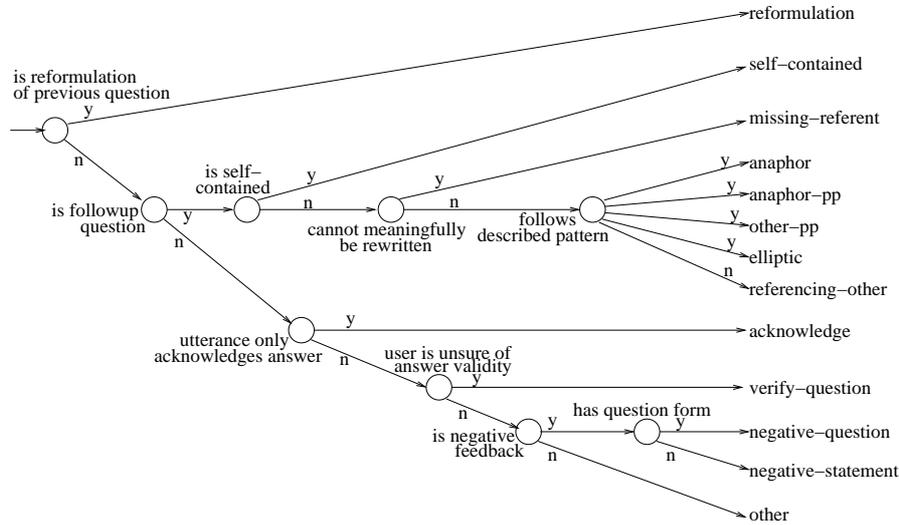
Figure 3. *Decision tree used in the annotation manual. Note that there are a couple of small differences with the classes described here. For the sake of simplicity we lumped together the classes negative-question and negative-statement into "negative". Also lumped together are the two classes anaphor and anaphor-pp (which distinguish two kinds of anaphor specifi c to the Dutch language).*

We found that, for about 80% of the fuu, an absolute majority of annotators agreed on the fuu's class. For our feasibility study, which requires classification accuracies of only about 50-80%, the noise implied in this result is not likely to be prohibitive. But, the less than perfect results do warrant an investigation into the validity of the classification. We determined the most frequent disagreements, and tried to analyse whether these were a result of inadequacies of the annotation procedure, or problems inherent to the classification itself. The coincidence matrices showed two kinds of problems: classes which were confused with multiple other classes, and disagreements between specific pairs of classes.

The most problematic classes were, not so surprisingly, missing-referent, and, more surprisingly, other-pp. We expected that a well-defined class like other-pp would not cause so many problems. Runners-up were (unsurprisingly) other and (surprisingly-again) anaphor. Particularly well-classified were negative and acknowledge. The most common inter-class confusions were anaphor/other-pp, anaphor/referencing-other, anaphor/missing-referent, reformulation/self-contained, and reformulation/negative.

We took a closer look at these particular classes and disagreements. Some clear conclusions could be drawn. We will look at the most important disagreements here.

– **missing-referent vs others.** Missing-referent was most frequently interchanged

with the anaphor, other-pp, and referencing-other classes. Disagreements between missing-referent and these others seemed to be based on two things: differences in the threshold at which annotators decided that the question was not practically rewritable, and differences in whether either the rewritability or the intent of the question was more important. There is some inherent vagueness inherent in these disagreements, as it does not seem clear how to draw the lines here.

– **anaphor vs other-pp**. This usually happened when the anaphor referred to a complex NP with a PP in it, implying that both methods could be applied with the same result. This problem may be easily solved by giving either priority over the other in the annotation scheme.

– **anaphor vs referencing-other.** These were typically about what method was found to rewrite the question. Some annotators did not find the anaphor pattern while it was actually applicable, others seemed to have disagreements about what exactly is a well-formulated self-contained question. Disagreements may be reduced by further specification and giving more examples, though some vagueness is inherent in the question of what forms a well-formulated fuq.

– **reformulation vs self-contained.** these seemed to be caused by disagreements of whether a question had the same content as the original question. This may be an inherent vagueness, and may even warrant deleting the reformulation class altogether (that is, merging reformulation and self-contained into one class).

– **reformulation vs negative (mostly negative questions).** Most disagreements found were actually about negative questions which were literal repetitions of the original question, but with a few stop words added. In fact, reformulation was obviously chosen more often than was meant. It should be easy to draw a clearer line between the two classes by more detailed specification and more examples.

We conclude that we can make some minor improvements in the annotation scheme that are likely to improve Kappa, and that some inherent vagueness remains, in particular the distinction between missing-referent and a "regular" fuq. Still, we consider our current classification satisfactory for our feasibility study. One thing that we decided after this analysis is to lump together the reformulation and self-contained classes.

5. Evaluation of dialogue engine

To investigate the possibility of classifying and handling our identified types of fuu by a dialogue system, we extended the QA dialogue engine currently in the IMIX demonstrator to include our fuu classification. This dialogue engine is being developed as part of the IMIX demonstrator, and also contains functionality outside of the scope of this paper. While it is still limited in certain aspects, the demonstrator is a fully-functional dialogue system that can be run stand-alone on a laptop or in a client-server configuration over the internet. The purpose of our implementation here is not to introduce new, state of the art, language technology, but to see if application

of proven and preferably broad-domain techniques are enough to get a serviceable dialogue manager.

The dialogue engine contains a fuu classifier, a reference resolver, and a query generator which handles the question rewriting. An incoming utterance is first classified, syntactically tagged, and its possible anaphoric references identified. The dialogue engine then decides whether the question is passed to the QA, or whether another action needs to be taken. Once passed to the QA, the question rewriting module generates the rewritten question and passes all dialogue hints to the QA. The three QA engines work in parallel, and produce zero or more answers. The dialogue engine then chooses which answer to display, or possibly, whether a different dialogue act is appropriate. We chose not to invoke the QA multiple times in a single turn, because of performance issues: QAs are typically slow; ours take between 2 and 30 seconds to answer. Instead, we decided that it is the QA's responsibility to provide an appropriate answer the first time round, and to implement any required iterations, such as request relaxation, by itself, and report relevant issues back to the dialogue manager. Some NLP tools readily available to us in the system were a POS tagger and domain-specific semantic tagger from the Rolaquad QA project (Lendvai, 2005), and the broad-coverage Alpino dependency tree parser (Bouma *et al.*, 2006).

Evaluating the impact of dialogue managent on QA performance is not easy. Different QAs with different document bases will have different performance. The three QAs that we have at our disposal still have a rather low practical performance, which will make their answers unusable for evaluation. Therefore we will not look at actual QA results here. Instead we will consider several separate things that can be evaluated without actually invoking a QA: correctness of classification, correctness of rewriting, and probability that a fuq can be found in the document that the previous answer came from. These results were obtained by having the dialogue manager run in "offline" mode, where utterances from a corpus can be run through the system as if they were real user input. This information will be combined to provide an estimate of the effectiveness of our dialogue manager.

5.1. *Classification performance*

We first implemented the classifier by manually selecting words, phrases, and POS tags, that would be cues for specific classes according to intuitive language theory, and which would give the best performance when tested on our corpus. We then compared this with machine learning classifiers, using all unigrams, bigrams, and trigrams of words and POS tags in the sentence as input features.

The manual approach has two advantages over machine learning: prevention of non-generalisable results or "overfitting" (since the corpus is really a bit small for machine learning), and obtaining insight into the language phenomena involved. If the manual classifier has a performance similar to an optimal machine learning algo-

rithm that uses a wide choice of features, this gives us an indication that our intuitive algorithm didn't miss something important.

With our manual algorithm, we obtained a performance of 55% over all utterances in the corpus. We used the Weka toolkit (Witten *et al.*, 2005) to try different machine learning algorithms. The best performance we found was in the 55%-60% range, very close to our manual algorithm. The different algorithms and feature selection filters also tended to select mostly the same features as our manually chosen ones. The binary tree classifier even came up with binary decision trees similar to our own. We managed to get a performance of 62%-63% by using support vector machines, and adding an extra feature which denoted whether the semantic classifier found a sufficiently specific medical domain term in the sentence.

This is a reasonable result for such a knowledge-poor approach. It is enough to start with, especially since the manual algorithm concentrates on avoiding false positives with respect to non-default behaviour, rather than optimising overall classification performance per se. For example, misclassifying an "anaphor" fuu as "negative" would be a disaster, while misclassifying "anaphor" as "referencing-other" only does limited damage. This ensures that most misclassifications are relatively safe, rather than costly, in terms of dialogue performance. In particular, referencing-other and self-contained were chosen as "sensible defaults" in case of uncertainty. If we consider all matches of self-contained and referencing-other as safe defaults for the utterances classified as fuq, our manual algorithm arrives at an accuracy of 73%.

In the rest of the section, we will describe the most important rules we used for classification. These rules are very simple, and most of them seem transferable to other languages.

The words *niet* (*not*) and *geen* (*none*), together with the utterance not being a question, detected some 80% of negative statements. Negative questions could be detected using *maar* (*but*) at the beginning of a sentence or phrase, or the occurrence of *of niet* (*isn't it*).

Some 75% of acknowledgements could be detected by the words *dank* (*thank*) and variants, *ok*, and *oh*, *jammer* (*a pity*), *duidelijk* (*that's clear*), *mooi* (*nice*); and *dan* (*so*) at the beginning.

A variety of special cue words could be used to detect some subclasses of difficult-to-rewrite sentences (which should be classified as referencing-other). In our corpus we found *zo'n* (*such a*) (indicating reference based on similarity, which we can't handle), *zoveel* (*that many*) (referring to quantities), *andere* (*other*) (referring to set operations).

We found that questions starting with the word *dus* (*so*) are almost always verify-questions.

The occurrence of certain wh-words at the end of a sentence (indicating a question written in statement form) was a good indicator of some of the cases of missing-

referent. In particular ... *wat ?* (... *what?*), or ... *waarvan ?* (... *of what?*), ... *waardoor ?* (... *by what?*).

For analysing anaphors, we looked at the presence of regular determiners, like *de* (*the*) and *deze* (*this*), and PP-type determiners, which are a specifically Dutch thing, for example, *erdoor* (*by it*), *hiermee* (*with this*), *daarvan* (*of that*). This detects most of the instances of the anaphor class along with the positions of the anaphors.

5.2. *Rewriting performance*

Attempts were made to rewrite the anaphor, elliptic, and PP attachment classes. We also looked at whether antecedents were found in the initial question or the answer. We found that antecedents were often found in *both* the question and the answer. We obtained the following results:

Anaphor. Anaphor proved to be the easiest, as the majority of these can be found using cue words and POS tags, while the antecedents can be found using a Lappin and Leass (Lappin *et al.*, 1994) type salience algorithm. Some simplifications could be made. In particular we found that the antecedent could be found in the question in some 75% of the cases, so we optimised our salience algorithm a bit by increasing the salience of antecedents found in the user utterance. Still, our achievements were limited. Of all fuqs labeled by the system as "anaphor", only 42% were rewritten properly. This was in part due to a large number of false positives and in part due to errors in the reference resolution.

Elliptic. Elliptic proved to be even more difficult. The classification performance was reasonable. We detected 86% of all fuq of class elliptic by just looking at the absence of a verb, with 44% false positives. Rewriting was done by finding the sentence that the elliptic expression referred to, and then using constituents from that sentence to form a self-contained sentence. Performance of finding these antecedent sentences and rewriting were unusably low, however. Just a small minority of the antecedent sentences can be found by matching the elliptic sentence with words from previous sentences. There were no easy shortcuts available either. We found that only 55% of the elliptic fuq referred exclusively to the original user question, and 14% referred to candidate sentences in both the user question and the system answer. Building a correct sentence from these proved difficult as well. Syntactic transformation using the Alpino dependency tree parser did not work in most cases. Our first attempts showed that the transformed sentences were unsyntactic or did not make sense. We also tried building sentences from relation-argument type semantic frames using the Rolaquad semantic tagger, but the tagger did not seem reliable enough to get useable results.

Other-PP. The other-pp class of rewritable questions are relatively difficult to recognise, as fuq which require a PP to be attached are not readily distinguishable from self-contained questions. An obvious approach is to use semantic knowledge in the form of verb-argument relations, as is used in PP attachment disambiguation (Gildea *et al.*, 2001). Again, however, our available semantic tagger did not prove

reliable enough for this. What we did find is that 62% of other-pps referred to an NP occurring in both user and system utterance, and a total of 89% referred to an NP occurring in the user utterance. This suggests that other-pp has at least potential for use in the IR context approach.

Our first conclusion is that rewriting is not easy, and we will need accurate low-noise domain-specific semantic knowledge to do it. We did not focus on such domain-specific techniques here, so our current system only fully handles the anaphor class. Alternatively, anaphor and other-pp can be used as an IR context indicator that the user utterance should be used.

5.3. *Potential performance of the IR context approach*

One of the most popular IR context approaches is to search only through the previous n documents retrieved by the previous question. De Boni reported a success rate of 50% for his own corpus (which is a combination of real-life and Trec dialogues). Why does this simple approach work so well? Intuitively, it is likely that most documents are coherent with respect to their topic, and that a single document is made so as to answer a number of common questions on that topic. In fact, this is underwritten by results from research by (Lin *et al.*, 2003). They studied answer length in relation to user preference and behaviour. They found that, within a specific information need scenario, using longer text fragments as answers produced dialogues with less questions.

To gain more insight into this most basic IR context approach, we evaluated the simplest version of it, namely only looking at the document the answer came from (be it correct or incorrect). This way, the result depends only on the nature of these specific documents, and not on the way in which the IR matches documents. The result we obtain may be considered a lower bound for the performance of the IR context approach with respect to document selection performance, and an upper bound with respect to the expected document fragment selection performance.

We checked manually whether the answer to each fuq in the corpus could be found in the document where the original answer came from. We only did this for the *correct* answers, since considering incorrect answers here introduces noise related to the performance of the specific IR used. That is, you will also be measuring the tendency of the actual IR to either select the wrong document, or the wrong sentence from the right document.

We checked all fuq, including self-contained fuq. For about 1/3 of the answers, the source document could not be retrieved because of errors or incompleteness in their source references. The documents were nearly all sections from encyclopedias, and ranged from 50-500 words in size, with an average of about 150. A total of 196 fuq were checked this way. As a way of indicating the existence of vagueness in the documents' answering potentials, each fuq was annotated with a 3-point scale, thus including a "partial match" option:

— no match: the document could not answer the question in any way.

— partial match: the document gave only a partial answer or an implicit answer.

— full match: a satisfactory answer was explicitly stated in the document.

We found that relatively few cases needed to be annotated as "partial match", so we considered only the distinction "full match" versus "no full match". We found that some 39% of the fuq could be answered by the document the answer came from. For the subclass missing-referent, this percentage was 73%. This high figure is consistent with the concept of missing-referent as directly referring to the document the answer came from. This means that fuq of this class can be effectively dealt with by directly showing more of the answer's source document.

For the remaining fuq classes, the percentage was 35% on average. Percentages for each class varied somewhat, ranging from about 20% to about 43%. The differences were not very significant, and in particular we did *not* find that self-contained fuq have a lower percentage of matches, in fact it was 43%. The lowest was elliptic, with 20% (3 out of 15).

This rather high figure may have some interesting implications. In fact, it is consistent with our stated intuition of document coherence being the real reason behind the success of this approach. Implementing a simple "use last document" strategy is likely to be worthwhile in any QA dialogue system, as long as there is a proper way to detect when the answer could not be found in the last document, and other strategies are available to complement it.

Another implication is that perhaps we should reconsider the ways in which to select a text fragment from a document. Our results suggest that including more text in the answer text fragments may lead to better satisfaction of the users' information needs. On the other hand, our users seemed to prefer relatively small text fragments (as some started making comments on the text size when these were more than 4-5 sentences long).

5.4. *Overall dialogue performance*

In this section, we describe a technique to estimate the overall performance impact of the dialogue handling of our dialogue system and proposed IR context scheme. We distinguish the following two tasks in dialogue management:

1) Identifying whether a fuu is a fuq or not, and whether it is an acknowledge, negative feedback, or verify-question.

2) In case the fuu is a fuq, passing the rewritten question and IR context hints to the QA, or producing the document in case the fuq is a missing-referent. For this task, we consider two cases:

a) QAs without any IR context abilities. We assume that the dialogue manager cannot pass IR context hints. The baseline is always passing the question as a self-

contained question.

b) QAs with IR context abilities. We assume the dialogue manager may pass IR context hints. The baseline is to consider all fuu to be fuq, with only the context flag set.

With respect to these tasks, we consider the cases in which our dialogue manager would provide better, equal, or worse results than the baseline, using the following criteria:

– **Better.** fuq is correctly rewritten; IR context is correctly specified; fuu is correctly classified as negative, acknowledge, or verify-question; fuq is correctly classified as missing-referent.

– **Same.** Behaves same as baseline.

– **Worse.** fuq is wrongly rewritten while the original fuq would be a better query to the QA, or the wrong IR context hints are passed; wrongly identifies missing-referent; wrongly identifies negative, acknowledge, or verify-question.

Our dialogue manager has the following general strategy:

– If fuu is negative-feedback, acknowledge, or verify-question, we prompt accordingly (we might use any system clarification or answer selection strategies if present, currently there are none).

– If fuu is missing-referent, we show the document that the answer came from.

– If fuu is another type of fuq, we pass to the QA: the IR context (either the last question or the last answer), and the rewritten question. In particular:

   - anaphor: we pass the IR context according to the predicted antecedent, and/or the rewritten question.

   - other: we pass no specific hints, just that the question is a fuq.

For task 1 we found the following results:

Better	negative, acknowledge, verify-question identified	138 (24%)
Same	all fuq identified as fuq	405 (70%)
Worse	wrongly identified a non-fuq class	32 (5.6%)

For task 2 we considered only the 405 utterances classified as fuq. We found the following for case (a) (QA without context abilities):

Better	missing-referent correctly identified	8 (2.0%)
	question correctly rewritten	37 (9.1%)
Same	original question passed to QA	405 (70%)
	question wrongly rewritten, original not self-contained	52 (13%)
Worse	question wrongly rewritten, original was self-contained	1 (0.2%)
	incorrectly identified missing-referent	1 (0.2%)

The system does marginally better (11%) by grace of the missing-referent handling, the fact that the QA has no alternative to the anaphor rewriting, and the fact that there are hardly any false positives where it produces worse dialogue behaviour.

For case (b), the QA can handle context on its own, but we are enabled to use IR contexts shortcuts where possible. We found that, in the anaphors we detected correctly, 89% referred to a concept in the user utterance. As a strategy, we can just pass a pointer to the user utterance instead of rewriting the sentence. This only gives us a dubious improvement of 1%, however. We would get:

Better	missing-referent correctly identified	8 (2.0%)
	IR context correctly specified	40 (10%)
Same	original question passed to QA	405 (70%)
Worse	IR context wrongly specified	49 (12%)

In other words, we do worse in as many cases as we do better. Finally, let's see how much better the QA in case (b) handles fuq, if we assume the IR context strategy we discussed in section 5.3. We found there that QAs are likely to find at least 35% of fuq by a default IR context approach, providing that the original answer is correct. For the non-self-contained fuq (75% of all fuq) this may give us an improvement of 35%, which amounts to an improvement of up to 26%, depending on how many of the answers are correct.

6. Conclusions

6.1. *Comparisons with other research*

While general fuu classifications are lacking in the literature, there are several other fuq classifications. Like ours, these are intuitive typologies, chosen for their possible usefulness for dialogue system implementation. However, the relationship between the classifications and a real implementation is typically not further developed, as ours is. We will distinguish two kinds of classifications: classifications of the discourse function of the fuq (or, the semantic-pragmatic function), and of the anaphoric relationship between fuq and previous utterances.

The discourse function classifications are typically about topic shifts. In (Chai *et al.*, 2004), for example, three main classes are proposed: topic extension, topic exploration, and topic shift. They relate it to a potential implementation using semantic frames, where the discourse function determines how semantic frames should be merged. (Bertomeu *et al.*, 2006) proposes a division into questions referring to previous questions and those referring to previous answers. Question-to-question types are refinement, theme-entity, theme-property, paraphrase, and overlap. Question-to-answer classes are refinement and theme. These classes can be related to how the query slots of a hypothetical IR system should be filled in with information from different utterances. Both classifications work at the semantic frame level, which we

chose not to model, because of its dependence on a specific IR system. Unfortunately, neither systems has been tested with an implementation.

The anaphoric relationship classifications are comparable to our fuq classification. (Bertomeu *et al.*, 2006) proposes a division into three kinds of anaphoric references: regular co-reference (which is mainly what we've looked at here), subset/superset relations, and bridging (i.e. referring to an entity that can only be inferred). It may be interesting to see to what extent the latter two classes can be used to subdivide our "referencing-other" class. Another classification, based on surface form of the utterance, is proposed in (Kato *et al.*, 2006): pronoun, zero pronoun, definite NP, ellipsis, and no reference expression. This classification is very similar to our own, even if the system is in Japanese. For both of these models, the relation of the classification to an actual rewritten sentence or IR query, and to an implementation, remain as yet unclear.

In our corpus, we found that 56% of utterances can be considered fuq. Of these, 25% are self-contained, and of the relevant non-self-contained ones, 63% are found to be potentially rewritable using machine-ready techniques. Of this 63% we managed to rewrite only 18% correctly with our first dialogue system. We also found that 39% of all fuq could be answered by just looking at the document that the previous answer came from.

There are few similar analyses of QA dialogue corpora in the literature. An important one is the work by De Boni (De Boni *et al.*, 2004). He found that 37% of fuq in his corpora could be answered as self-contained questions, and 69% of fuq could be solved by either considering them self-contained, or looking at the documents from the initial question. Of the remaining 31%, 45% (14% of total) could potentially be solved with anaphora resolution. These figures seem reasonably consistent with our own findings. Note that De Boni did not provide algorithms for rewriting or for distinguishing between the different sentence classes necessary to do this; he elaborates only the distinction self-contained versus context-needed, a distinction not addressed in this research.

A number of QA dialogue systems are described in the literature, but of few of them, performance figures are given, perhaps due to the lack of coverage by QA contests. The Rits-QA system does mention a figure, it uses three rewriting strategies similar to our own, to rewrite 37% of the sentences in their corpus correctly. Other systems only had limited results, such as the 1% improvement found by the Alicante system (González *et al.*, 2000). While a few semantic frame based QA dialogue systems exist, no performance figures are given there.

6.2. *General conclusions*

We summarise our findings in a set of general conclusions here.

– A QA independent dialogue manager can be built. At least modest results could be obtained using simple and even domain-independent techniques.

– Distinguishing between fuq and other kinds of utterances (in particular, acknowledge and negative feedback) has considerable added value in a real-life dialogue system.

– Both the rewriting and IR context strategies we considered have added value, but we did not manage to provide cumulative added value of both.

- Anaphor resolution (using standard techniques) in particular seems to be feasible, even though our result (only 42% handled correctly) is limited.

- The simple "look in the last document only" strategy has significant added value, *providing* the system can detect sufficiently well whether the answer is in the last document. This finding also suggests using a different strategy for extracting answers from a document.

– Our fuu analysis is still limited by the lack of longer dialogues in our "second utterance" corpus, which only contains user utterances in the second user turn. While we have shown that we can use this corpus to analyse many important aspects of QA dialogue handling, we could in particular not analyse topic changes and longer-distance references.

– We found in several cases that we are in need of reliable semantic knowledge to improve our results further.

7. References

Akker R. o. d., Bunt H., Keizer S., Schooten B. v., " From question answering to spoken dialogue: towards an information search assistant for interactive multimodal information extraction", *Interspeech 2005*, p. 2793-2796, 2005.

Bertomeu N., Uszkoreit H., Frank A., Krieger H.-U., Jörg B., " Contextual phenomena and thematic relations in database QA dialogues: results from a Wizard-of-Oz Experiment", *Workshop on Interactive Question Answering, HLT-NAACL 06*, 2006.

Bouma G., Mur J., van Noord G., van der Plas L., Tiedemann J., " Question Answering for Dutch using Dependency Relations", *Proceedings of the CLEF2005 workshop*, 2006.

Chai J. Y., Jin R., " Discourse Structure for Context Question Answering", *HLT-NAACL QA workshop 2004*, 2004.

Cohen J., " A coefficient of agreement for nominal scales", *Educational and Psychological Measurement*, vol. 20, p. 37-46, 1960.

De Boni M., Manandhar S., " Implementing Clarification Dialogues in Open Domain Question Answering", *Journal of Natural Language Engineering*, 2004.

Fukumoto J., " Answering questions of Information Access Dialogue (IAD) task using ellipsis handling of follow-up questions", *Workshop on Interactive Question Answering, HLT-NAACL 06*, 2006.

Fukumoto J., Niwa T., Itoigawa M., Matsuda M., " RitsQA: List answer detection and Context task with ellipses handling", *Working notes of the Fourth NTCIR Workshop Meeting*, p. 310-314, 2004.

Galibert O., Illouz G., Rosset S., " Ritel: an open-domain, human-computer dialog system", *Interspeech 2005*, p. 909-912, 2005.

Gildea D., Palmer M., " The Necessity of Parsing for Predicate Argument Recognition", *Proceedings of the 40th Annual Meeting on Association for C omputational Linguistics*, Annual Meeting of the ACL, Philiadelphia, 2001.

González J. L. V., Rodríguez A. F., " A Semantic Approach to Question Answering Systems.", *TREC*, 2000.

Herrera J., Peñas A., Verdejo F., " Question answering pilot task at CLEF 2004", *Multilingual Information Access for Text, Speech and Images: 5th Workshop of the Cross-Language Evaluation Forum, CLEF 2004*, p. 581-590, 2005.

Inui K., Yamashita A., Matsumoto Y., " Dialogue management for language-based information seeking", *Proc. First International Workshop on Language Understanding and Agents for Real World Interaction*, p. 32-38, 2003.

Kato T., Fukumoto J., Masui F., Kando N., " WoZ Simulation of Interactive Question Answering", *Workshop on Interactive Question Answering, HLT-NAACL 06*, 2006.

Krippendorff K., *Content Analysis: An Introduction to Its Methodology*, Sage Publications, Beverly Hills, CA, 1980.

Lappin S., Leass H. J., " An Algorithm for Pronominal Anaphora Resolution", *Computational Linguistics*, vol. 20, n° 4, p. 535-561, 1994.

Lendvai P., " Conceptual taxonomy identification in medical documents", *Proceedings of The Second International Workshop on Knowledge Discovery and Ontologies (KDO-2005)*, p. 31-38, 2005.

Lin J., Quan D., Sinha V., Bakshi K., Huynh D., Katz B., Karger D. R., " What makes a good answer? The role of context in question answering", *Proceedings of the Ninth IFIP TC13 International Conference on Human-Computer Interaction (INTERACT-2003)*, 2003.

Nyberg E., Frederking R., Mitamura T., Bilotti M., Hannan K., Hiyakumoto L., Ko J., Lin F., Lita L., Pedro V., Schlaikjer A., " JAVELIN I and II Systems at TREC 2005", *Proceedings of TREC 2005*, 2005.

Oh J.-H., Lee K.-S., Chang D.-S., Seo C. W., Choi K.-S., " TREC-10 Experiments at KAIST: Batch Filtering and Question Answering.", *TREC*, 2001.

Reithinger N., Bergweiler S., Engel R., Herzog G., Pfleger N., Romanelli M., Sonntag D., " A look under the hood: design and development of the first SmartWeb system demonstrator", *ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces*, ACM Press, New York, NY, USA, p. 159-166, 2005.

Tjong Kim Sang E., Bouma G., de Rijke M., " Developing Offline Strategies for Answering Medical Questions", *Proceedings of the AAAI-05 Workshop on Question Answering in Restricted Domains*, p. 41-45, 2005.

Voorhees E. M., " Overview of TREC 2001.", *TREC*, 2001.

Witten I. H., Frank E., *Data Mining: Practical machine learning tools and techniques, 2nd Edition*, Morgan Kaufmann, 2005.