Learning to Classify Non Sentential Utterances

Raquel Fernández

Group of Logic, Language & Computation
Department of Computer Science
King's College London

raquel@dcs.kcl.ac.uk

[Joint work with Jonathan Ginzburg & Shalom Lappin]

Non Sentential Utterances

Conversations are full of fragmentary utterances
- at least 11% (Fernández and Ginzburg 2002)

Anon 1: How did you travel?
Anon 2: **By, by bus**. [BNC, HDM]

. . .

Kay: That's no good because the mass is on Thursday.
Unknown: $<whispering>$ **Which Thursday?**
Kay: **This Thursday coming**. [BNC, HDD]

. . .

Janet: Right, David I'll talk to you.
Carol: **About?**
Janet: **Erm, unit linking**. [BNC, JK8]

Non Sentential Utterances: fragmentary form but full sentential meaning

Non Sentential Utterances

- NSUs are a challenging problem both for linguistic theories and implemented dialogue systems
- Arguably the most important issue in the processing of NSUs concerns their **resolution** (i.e. the recovery of a clausal meaning from an incomplete form)
- However, a necessary first step towards this final goal is the identification of the right NSU type, which will determine the appropriate resolution procedure
- Here we address this latter issue, namely the **classification** of NSUs, using a machine learning approach.

Talk Overview

- Taxonomy of NSUs
- Resolving NSUs - theory sketch
- Learning to classify NSUs
  - Data
  - Features
  - Baseline
  - Machine learning results
- Evaluation & discussion

NSU Taxonomy

- We follow the corpus-based taxonomy of NSUs developed in (Fernández & Ginzburg 2002)
- It was designed after exhaustive analysis of 10 randomly chosen BCN files, assisted by the search engine SCoRE (Purver 2001).
- It was tested by manual annotation of a randomly selected section of 200-speaker-turns from 54 BNC files. The examined sub-corpus contains 14,315 sentences.
- We found a total of 1296 NSUs. They were all labeled according to the typology presented below, and annotated with the sentence number of its antecedent utterance.
- Annotation: manual using decision trees. Reliability: in progress.

NSU Taxonomy

- The original taxonomy contains a total of 15 classes
- Here we focus on 13 classes, excluding plain acknowledgments (like "Mhm") and check questions (like "Okay?")
- We partition NSUs into *question-denoting* types and *proposition-denoting* types

NSU Taxonomy - Question-denoting NSUs

- Sluices and Clarification Ellipsis (CE) are the two classes of NSUs that denote questions.
- **Sluices**: We consider as sluices all *wh*-question NSUs (contra F&G 2002's taxonomy), thereby conflating *direct* with *reprise* sluices. (Fernández et al. 2004) show that sluice interpretations can be efficiently disambiguated using machine learning techniques.

(1) June: Only wanted a couple weeks.
Ada: What? [KB1, 3312]

(2) Cassie: I know someone who's a good kisser.
Catherine: Who? [KP4, 512]

NSU Taxonomy - Question-denoting NSUs

- **Clarification Ellipsis (CE)** We use this category to classify reprise fragments used to clarify an utterance that has not been fully comprehended.

  (3) A: There's only two people in the class
  B: Two people? [KPP, 352–354]

  (4) A: . . . You lift your crane out, so this part would come up.
  B: The end? [H5H, 27–28]

NSU Taxonomy - Proposition denoting NSUs

- **Short Answer**: Short Answers are typical responses to (possibly embedded) *wh*-questions.

  (5) A: Who's that?
  B: My Aunty Peggy. [G58, 33–35]

  (6) A: Can you tell me where you got that information from?
  B: From our wages and salary department. [K6Y, 94–95]

- However, there is no explicit *wh*-question in a short answer to a CE question (7), nor in cases where the *wh*-phrase is ellided (8).

  (7) A: Vague and?
  B: Vague ideas and people. [JJH,65–66]

  (8) A: What's plus three time plus three?
  B: Nine.
  A: Right. And minus three times minus three?
  B: Minus nine. [J91, 172–176].

NSU Taxonomy - Proposition denoting NSUs

- **Plain Affirmative Answer and Rejection**: The typical context of these two classes of NSUs is a polar question.

  (9) A: Did you bring the book I told you?
  B: Yes./ No.

- They can also answer *implicit* polar questions, e.g. CE questions:.

  (10) A: That one?
  B: Yeah. [G4K, 106–107]

- Rejections can also be used to respond to assertions:

  (11) A: I think I left it too long.
  B: No no.[G43, 26–27]

- Both plain affirmative answers and rejections are strongly indicated by lexical material, characterized by the presence of a "yes" word ("yeah", "aye", "yep"...) or the negative interjection "no"

NSU Taxonomy - Proposition denoting NSUs

- **Repeated Affirmative Answer** Typically, repeated affirmative answers are responses to polar questions. They answer affirmatively by repeating a fragment of the query.

  (12) A: Did you shout very loud?
  B: Very loud, yes. [JJW, 571-572]

- **Repeated Acknowledgment** This class is used for acknowledgments that, as repeated affirmative answers, also repeat a part of the antecedent utterance, which in this case is a declarative.

  (13) A: I'm at a little place called Ellenthorpe.
  B: Ellenthorpe. [HV0, 383–384]

NSU Taxonomy - Proposition denoting NSUs

- **Helpful Rejection** The context of helpful rejections can be either a polar question or an assertion.
- In the first case, they are negative answers that provide an appropriate alternative:

  (14) A: Is that Mrs. John [*last or full name*]?
  B: No, Mrs. Billy. [K6K, 67-68]

- As responses to assertions, they correct some piece of information in the previous utterance:

  (15) A: Well I felt sure it was two hundred pounds a, a week.
  B: No fifty pounds ten pence per person. [K6Y, 112–113]

NSU Taxonomy - Proposotion denoting NSUs

- **Propositional and Factual Modifiers** These two NSU classes are used to classify propositional adverbs like (16) and factual adjectives like (17), respectively, in stand-alone uses.

  (16) A: I wonder if that would be worth getting?
  B: Probably not. [H61, 81–82]

  (17) A: So we we have proper logs? Over there?
  B: It's possible.
  A: Brilliant! [KSV, 2991–2994]

- **Bare Modifier Phrase** This class refers to NSUs that behave like adjuncts modifying a contextual utterance. They are typically PPs or AdvPs.

  (18) A: . . . they got men and women in the same dormitory!
  B: With the same showers! [KST, 992–996]

NSU Taxonomy - Proposotion denoting NSUs

- **Conjunction + fragment** This NSU class is used to classify fragments introduced by conjunctions.

  (19) A: Alistair erm he's, he's made himself coordinator.
  B: And section engineer. [H48, 141–142]

- **Filler** Fillers are NSUs that fill a gap left by a previous unfinished utterance.

  (20) A: [. . .] twenty two percent is er <*pause*>
  B: Maxwell. [G3U, 292–293]

Resolving NSUs - Theory

- (Ginzburg & Sag 2001) analyse NSUs by combining

  HPSG - grammatical constructions (Sag 1997)

  KOS - a theory of context in dialogue (Ginzburg 1996, forth.)

- The main idea is that NSUs get their main predicates from context, via combination (unification or functional composition) with the question that is currently under discussion

$$\begin{bmatrix} \text{MAX-QUD} & \textit{maximal question under discussion} \\ \text{SAL-UTT} & \textit{salient utterance} \end{bmatrix}$$

NSU Resolution - the SAL-UTT

- The *salient-utterance* parameter (SAL-UTT) represents the antecedent sub-utterance. This plays a role similar to the *parallel element* in higher order unification–based approaches (see e.g. Dalrymple et al. 1991 and Pulman 1997)
- The SAL-UTT provides a partial specification of the *focal (sub)utterance* - it is computed as the (sub)utterance associated with the role bearing widest scope within MAX-QUD.
- SAL-UTT is used to encode *syntactic and phonological parallelism* between the fragment and an antecedent (e.g. case matching or even phonological identity).
- Many of the heuristics we develop for disambiguating NSUs relate to identifying the SAL-UTT.

NSU Resolution - Identifying MAX-QUD

- In the most prototypical case, the MAX-QUD is the content of the *most recent utterance*, like in **short answers**; the SAL-UTT in such a case is the sub-utterance of the wh-phrase:

  A: Who phoned? B: Bo (= Bo phoned).

  MAX-QUD: $\lambda x.Phone(x,t)$

- For **propositional lexemes** such as 'yes', 'no', and 'probably' the MAX-QUD is a polar question $p?$ such that the (content of the) most recent utterance is (an assertion of) p.

  A: Did Bo phone?
  B: Yes/No/Probably (= Bo phoned/didn't phoned/probably phoned).

  A: Bo phoned.
  B: Yes/No/Probably (= Bo phoned/didn't phoned/probably phoned).

  MAX-QUD: $?Phone(b,t)$

NSU Resolution - Identifying MAX-QUD

- In **direct sluicing**, the MAX-QUD is a polar question $p?$, where p is required to be a quantified proposition; the SAL-UTT in such a case is the sub-utterance associated with the widest scoping quantifier:

  A: A student phoned. B: Who? (= Which student phoned?)

  A: Did someone phone? B: Who? (= Who phoned?).

  MAX-QUD: $?\exists x.Phone(x,t)$

- **Adjuncts sluices** are possible even without an overt antecedent; in such cases the value of SAL-UTT needs to be null

  A: John saw Mary.

  B: Why?/With who? (= Why/With who did John see Mary?)

  MAX-QUD: $?See(j,m,t)$

NSU Resolution - Identifying MAX-QUD

- MAX-QUD can also arise in a somewhat less 'direct' way, via a process of *utterance coercion* (see Cooper & Ginzburg 2002), triggered by the inability to *ground* (Clark 1996, Traum 1994) the previous utterance.
- In **reprise sluicing** and **CE**, the MAX-QUD is a question about the content of a sub-utterance which the addressee cannot resolve - the output of a coercion operation. The unresolved sub-utterance constitutes the SAL-UTT.

  A: Did Bo leave?

  B: Who? (= Who are you asking if s/he left?)

  MAX-QUD: $\lambda bAsk(A, ?leave(b, t))$

NSU Resolution - Implementation

- SHARDS (Ginzburg et al. 2001, Fernández et al. in press) is an implemented system which provides a procedure for computing the interpretation of NSUs in dialogue. The system comprises two main components: an HPSG-based grammar and a resolution procedure.
- The system currently handles short answers, direct and reprise sluices, as well as plain affirmative answers to polar questions.
- SHARDS has been extended to cover several types of clarification requests and used as a part of the information-state-based dialogue system CLARIE (Purver 2004a, Purver 2004b). In particular, CLARIE can parse and generate reprise sluices by implementing the aforementioned analysis of grounding/clarification interaction.

Talk Overview

- Taxonomy of NSUs
- Resolving NSUs - theory sketch
- Learning to classify NSUs
  - Data
  - Features
  - Baseline
  - Machine learning results
- Evaluation & discussion

Classifying NSUs - Data

The data used in our experiments was selected from our corpus of NSUs following two simplifying restrictions:

- We decided to leave aside *feedback* NSUs, i.e. those NSUs classified as Acknowledgments (595 instances) or as Check Questions (22 instances). We plan to incorporate them in a future phase of our investigation.
- We restrict our experiments to those NSUs whose antecedent is the *immediately preceding utterance*. This restriction, which makes the feature annotation task easier, does not pose a coverage problem, given that the immediately preceding utterance is the antecedent for the vast majority of NSUs (88%).

Classifying NSUs - Data

- Distribution of classes in the sub-corpus used in our experiments:

NSU class	**Total**
Short Answer	105
Affirmative Answer	100
Repeated Ack.	80
CE	65
Rejection	48
Repeated Aff. Ans.	25
Factual Modifier	23
Sluice	20
Helpful Rejection	18
Filler	16
Bare Mod. Phrase	11
Propositional Modifier	10
Conjunction + frag	5
Total dataset	**527**

Classifying NSUs - Features

We identify three aspects that play an important role in the NSU classification task:

- The first one has to do with semantic, syntactic and lexical properties of the NSUs themselves.
- The second one refers to the properties of its antecedent utterance.
- The third concerns relations between the antecedent and the fragment.

Classifying NSUs - Features

feature	description	values
ant_mood	mood of the antecedent utterance	decl,n_decl
wh_ant	presence of a *wh* word in the antecedent	yes,no
finished	(un)finished antecedent	fin,unf
nsu_cont	content of the NSU (either prop or question)	p,q
wh_nsu	presence of a *wh* word in the NSU	yes,no
aff_neg	presence of a "yes" / "no" word in the NSU	yes,no,e(mpty)
lex	presence of different lexical items in the NSU	p_mod,f_mod,mod,conj,e(mpty)
repeat	repeated words in NSU and antecedent	0-3
parallel	repeated tag sequences in NSU and antecedent	0-3

Classifying NSUs - Feature Annotation

- We annotate the 527 instance sub-corpus of NSUs with the features shown above
- Our *feature annotation algorithm* is similar to the one used in (Fernández et al. 2004), which exploits the SGML markup of the BNC
- All feature values are extracted automatically using the PoS information encoded in the BNC markup. The BNC was automatically annotated with a set of 57 PoS codes (known as the C5 tagset), plus 4 codes for punctuation tags, using the CLAWS system (Garside 1987)

Classifying NSUs - Feature Annotation

- Some features, like `nsu_cont` and `ant_mood` for instance, are *high level* features with no straightforward POS tag correlate. Punctuation tags (corresponding to intonation in speech) help to extract the values of these features, but the correspondence is still not unique.
- For this reason we evaluate our automatic feature annotation procedure against a small sample of *manually annotated data*.
- We randomly selected 10% of our dataset (52 instances) and extracted the feature values manually. In comparison with this gold standard, our automatic feature annotation procedure achieves 89% accuracy.
- We use only *automatically annotated data* for the learning experiments.

Learning to Classify NSUs - Experiments

- The experiments performed on the data set involve the following steps:
  - determining a *baseline*
  - running several *machine learning algorithms* on the data set
  - *evaluating the results* by comparison with the baseline
- All results reported were obtained by performing 10-fold cross-validation.
- The results will be presented as follows...

Learning to Classify NSUs - Experiments

- The tables show the recall, precision and f-measure for each class.

$$\text{recall} = \frac{\text{OK by system}}{\text{total in data}} \qquad \text{prec} = \frac{\text{OK by system}}{\text{total by system}} \qquad \text{f} = \frac{(\beta^2 + 1)\text{pr}}{\beta^2\text{p} + \text{r}}$$

- To calculate the *overall performance of the algorithm*, we normalize those scores according to the relative frequency of each class.
- The weighted overall recall, precision and f-measure, shown in the bottom row of the tables, is then the sum of the corresponding weighted scores.

NSU class	**recall**	**prec**	**f1**
class1	—	—	—
class2	—	—	—
Weighted Score	—	—	—

Baseline

Majority Class baseline

- The simplest baseline we can consider is to always predict the majority class in the data, in our case Short Answer.
- This yields a **6.7%** weighted f-score.

NSU class	**recall**	**prec**	**f1**
ShortAns	100.00	20.10	33.50
Weighted Score	**19.92**	**4.00**	**6.67**

Baseline

One Rule baseline

- A slightly more interesting baseline can be obtained by using a one-rule classifier, which chooses the feature which produces the minimum error.
- This creates a single rule which generates a decision tree where the root is the chosen feature and the branches correspond to its different values. The leaves are then associated with the class that occurs most often in the data, for which that value holds.
- We use the implementation of a one-rule classifier provided in the Weka toolkit (Witten & Frank 2000).

Baseline

One Rule baseline

- In our case, the feature with the minimum error is `aff_neg`. It produces the following one-rule decision tree, which yields a **32.5%** weighted f-score.

```
aff_neg:
   yes   ->   AffAns
   no    ->   Reject
   e     ->   ShortAns
```

NSU class	**recall**	**prec**	**f1**
ShortAns	95.30	30.10	45.80
AffAns	93.00	75.60	83.40
Reject	100.00	69.60	82.10
Weighted Score	**45.93**	**26.73**	**32.50**

Baseline

Four Rule baseline

- We consider a more substantial baseline that uses the combination of four features that produces the best results.
- The four-rule tree is constructed by running the J4.8 classifier (Weka's implementation of the C4.5 system) with all features and extracting only the four first features from the root of the tree.
- This creates a decision tree with four rules, one for each feature used, and nine leaves corresponding to nine different NSU classes.

Baseline

Four Rule baseline

```
nsu_cont:
 q ->   nsu_wh:
          yes     ->   Sluice
          no      ->   CE
 p ->   lex:
          conj    ->   ConjFrag
          p_mod   ->   PropMod
          f_mod   ->   FactMod
          mod     ->   BareModPh
          e       ->   aff_neg:
                         yes   ->   AffAns
                         no    ->   Reject
                         e     ->   ShortAns
```

Baseline

Four Rule baseline

- This four-rule baseline yields a **62.33%** weighted f-score.

NSU class	**recall**	**prec**	**f1**
CE	96.97	96.97	96.97
Sluice	100.00	95.24	97.56
ShortAns	94.34	47.39	63.09
AffAns	93.00	81.58	86.92
Reject	100.00	75.00	85.71
PropMod	100.00	100.00	100.00
FactMod	100.00	100.00	100.00
BareModPh	80.00	72.73	76.19
ConjFrag	100.00	71.43	83.33
Weighted Score	**70.40**	**55.92**	**62.33**

Machine Learners

- We are interested in comparing the results obtained using different learning algorithms. We use *four different machine learners*, which implement *three different learning strategies*.
- Two of the learners are based on the C4.5 decision tree algorithm:
  – **Weka's J4.8** (Witten & Frank 2000), and
  – the rule induction learner **SLIPPER** (Cohen & Singer 1999).
- We compare the results of these two systems with two other learners that use different learning methods:
  – a memory-based algorithm **TiMBL** (Daelemans 2003), and
  – a **maximum entropy** algorithm developed by Zhang Le (Le 2003).
- They are all well established, freely available systems.

Machine Learners: C4.5-based systems

- **The J4.8 decision tree learner** is an implementation of a slightly revised version of the popular C4.5 algorithm called Revision 8.

  The Weka toolkit provides a friendly interface which allows for the visualization of the output decision tree.

- **SLIPPER** uses iterative pruning and confidence-rated boosting to create a weighted rule set.

  We use the option `unordered`, which finds a rule set that separates each class from the remaining classes, giving rules for each class. This yields slightly better results than the default setting.

  Unfortunately, it is not possible to access the output rule set when cross-validation is performed.

Machine Learners: TiMBL

- As with all memory-based learning algorithms, **TiMBL** computes the similarity between a new test instance and the training instances stored in memory using a distance metric.
- As a distance metric we use the *modified value difference metric*, which performs better than the default setting (*overlap metric*).
- In light of recent studies (Daelemans 2002), it is likely that the performance of TiMBL could be improved by a more systematic optimization of its parameters, as e.g. in the experiments presented in (Gabsdil & Lemon 2003). Here we only optimize the distance metric parameter and keep the default settings for the number of nearest neighbors and feature weighting method.

Machine Learners: Maximum Entropy

- Our final experiment used a **maximum entropy algorithm**, which computes the model with the highest entropy of all models that satisfy the constraints provided by the features.
- The maximum entropy toolkit we use allows for several options. In our experiments we use 40 iterations of the default L-BFGS parameter estimation (Malouf 2002).

Machine Learners: Results

- Although the classification algorithms used implement different machine learning techniques, they all yield very similar results, around an **87% weighted f-score**.

- The maximum entropy model peforms best, although the difference between its results and those of the other algorithms is not statistically significant.

Results - SLIPPER

NSU class	recall	prec	f1
CE	93.64	97.22	95.40
Sluice	96.67	91.67	94.10
ShortAns	83.93	82.91	83.41
AffAns	93.13	91.63	92.38
Reject	83.60	100.00	91.06
RepAffAns	53.33	61.11	56.96
RepAck	85.71	89.63	87.62
HelpReject	28.12	20.83	23.94
PropMod	100.00	90.00	94.74
FactMod	100.00	100.00	100.00
BareModPh	100.00	80.56	89.23
ConjFrag	100.00	100.00	100.00
Filler	100.00	62.50	76.92
Weighted Score	**86.21**	**86.49**	**86.35**

Results - Weka's J48

NSU class	recall	prec	f1
CE	97.00	97.00	97.00
Sluice	100.00	95.20	97.60
ShortAns	89.60	82.60	86.00
AffAns	92.00	95.80	93.90
Reject	95.80	80.70	87.60
RepAffAns	68.00	63.00	65.40
RepAck	85.00	89.50	87.20
HelpReject	22.20	33.30	26.70
PropMod	100.00	100.00	100.00
FactMod	100.00	100.00	100.00
BareModPh	80.00	100.00	88.90
ConjFrag	100.00	71.40	83.30
Filler	56.30	100.00	72.00
Weighted Score	**87.62**	**87.68**	**87.29**

Results - TiMBL

NSU class	recall	prec	f1
CE	94.37	91.99	93.16
Sluice	94.17	91.67	92.90
ShortAns	88.21	83.00	85.52
AffAns	92.54	94.72	93.62
Reject	95.24	81.99	88.12
RepAffAns	63.89	60.19	61.98
RepAck	86.85	91.09	88.92
HelpReject	35.71	45.24	39.92
PropMod	90.00	100.00	94.74
FactMod	97.22	100.00	98.59
BareModPh	80.56	100.00	89.23
ConjFrag	100.00	100.00	100.00
Filler	48.61	91.67	63.53
Weighted Score	**86.71**	**87.25**	**86.66**

Results - Maximum Entropy

NSU class	recall	prec	f1
CE	96.11	96.39	96.25
Sluice	100.00	95.83	97.87
ShortAns	89.35	83.59	86.37
AffAns	92.79	97.00	94.85
Reject	100.00	81.13	89.58
RepAffAns	68.52	65.93	67.20
RepAck	84.52	81.99	83.24
HelpReject	5.56	77.78	10.37
PropMod	100.00	100.00	100.00
FactMod	97.50	100.00	98.73
BareModPh	69.44	100.00	81.97
ConjFrag	100.00	100.00	100.00
Filler	62.50	90.62	73.98
Weighted Score	**87.11**	**88.41**	**87.75**

Evaluation and Discussion of Results

- The four-rule baseline algorithm discussed above yields a **62.33%** weighted f-score. Our best result, the **87.75%** weighted f-score obtained with the maximal entropy model, shows a *25.42% improvement* over the baseline system.

System	**w. f-score**
Majority class baseline	6.67
One rule baseline	32.50
Four rule baseline	62.33
SLIPPER	86.35
TiMBL	86.66
J4.8	87.29
Maximal Entropy	87.75

Evaluation and Discussion of Results

- It is interesting to note that the four-rule baseline achieves very high f-scores with Sluices and CE—around **97%**. Such results are not improved upon by the more sophisticated learners.
- This indicates that the features `nsu_cont` and `nsu_wh` used in the four-rule tree are sufficient indicators to classify question-denoting NSUs.
- The same applies to the classes Propositional Modifier and Factual Modifier. The baseline already gives f-scores of **100%**.
- This is in fact not surprising, given that the disambiguation of these categories is tied to the presence of particular *lexical items* that are relatively easy to identify.

Evaluation and Discussion of Results

- Affirmative Answers and Short Answers achieve high recall scores with the baseline systems (more than **90%**).
- In the three baselines considered, Short Answer acts as the *default category*. Therefore, even though the recall is high (given that Short Answer is the class with the highest probability), precision tends to be quite low.
- By using features that help to identify other categories with the machine learners we are able to improve the precision for Short Answers by around **36%**, and the precision of the overall classification system by almost **33%**—from 55.90% weighted precision obtained with the four-rule baseline, to the 88.41% achieved with the maximal entropy model.

Evaluation and Discussion of Results

- The class with the lowest scores is clearly Helpful Rejection. TiMBL achieves a **39.92%** f-measure for this class. The maximal entropy model, however, yields only a **10.37%** f-measure.

- This shows that the feature `parallel`, introduced to identify this type of NSUs, is not a good enough cue.

Conclusions and Future Work

We have presented a machine learning approach to the problem of Non Sentential Utterance classification in dialogue.

- We have described a procedure for predicting the appropriate NSU class from a fine-grained typology of Non Sentential Utterances, using a set of automatically annotated features.
- We have employed a series of simple *baseline methods* for classifying NSUs in our data set extracted from the BNC. The most successful of these methods uses a decision tree with four rules and gives an f-score of **62.33%**.
- We then applied *four machine learning algorithms* to this data set and obtained an f-score of approximated **87%** for all of them.

Conclusions and Future Work

- This improvement, taken together with the fact that the four algorithms achieve very similar results suggests that our *features* offer a reasonable basis for machine learning acquisition of the typology adopted.
- In future work we will integrate our NSU classification techniques into an *Information State-based dialogue system* (SHARDS/CLARIE) that assigns a full sentential reading to fragment phrases in dialogue.
- This will require a *refinement of our feature extraction* procedure. It will not be restricted solely to PoS input, but will also benefit from other information that our system generates, such as dialogue history and intonation.