Parsing and Disfluency Modeling

Matthew Lease

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-05-15
October 2005

Parsing and Disfluency Modeling

Matthew Lease

Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912-1910 USA
mlease@cs.brown.edu

September 7, 2005

Abstract

We propose to extend research on parsing and disfluency modeling begun at the 2005 Summer Workshop on Language Engineering at Johns Hopkins University. The proposed research includes: (1) establishing a common definition of fillers across SimpleMDE and tree annotation guidelines, (2) investigating an alternative syntactic representation of fillers for more accurate parsing and more meaningful parser evaluation, (3) improving disfluency modeling by revising and extending existing noisy-channel based approaches, and (4) initiating cross-linguistic study of the interaction between parsing and disfluency modeling.

1 Introduction

At the 2005 Summer Workshop on Language Engineering at Johns Hopkins University (JHU-SWLE-2005)[1], the Parsing And Spoken Structural Event Detection (PASSED) project investigated the interaction between metadata (i.e. structural event) detection and parsing, measuring the degree to which greater knowledge of one could improve accuracy of the other. One part of this work evaluated the downstream effect of sentence-like unit (SU) boundary and edit word detection errors on parse accuracy, as measured on both manually transcribed and automatically recognized words. In the other direction, the project also explored use of syntactic features to improve SU boundary detection.

This proposal primarily addresses yet another component of the PASSED project: studying the interaction between parsing and *filler* detection. Relative to other areas of the PASSED project, work on this study began relatively late and produced only preliminary results, leaving the study largely unfinished. As such, a key component of this proposal is completing the study to the extent that general findings can be disseminated to the research community.

[1] http://www.clsp.jhu.edu/ws2005

Our motivation for studying filler detection and its relationship with parsing arises from several sources. First of all, it was shown several years ago that the presence of fillers significantly reduces parsing accuracy, as measured on the Switchboard treebank of English conversational telephone speech [1]. Given that there has been great interest in showing better repair detection leads to better parsing [2, 3, 4], one would expect a similar strategy could be employed to boost parsing accuracy via better filler detection. Surprisingly, no such experiment has yet been performed. In a different line of research, interest in filler detection also arose in the DARPA *Effective, Affordable, Reusable Speech-to-Text* (EARS) program[2], which identified filler detection as a subtask of *rich transcription* and sponsored SimpleMDE guideline creation [5], data annotation, and community challenge evaluations. While both of these sources motivated interest in filler detection, the convergence of rich transcription and parsing in the PASSED project has provided a tremendous additional motivation: the availability for the first time of data dually annotated under both SimpleMDE and treebanking annotation standards[3]. As a final piece of motivation, a preliminary experiment we have run has confirmed that a modern parsing model [8] is struggling at filler detection. In this experiment, a parse tree is input to a simple rule-based filler detection system which labels as a filler any word that both (1) is known to commonly occur as a filler and (2) descends from INTJ or PRN in the input tree[4]. On human annotated gold trees this simple system achieves 11.6% error[5], but on parser output this error rate doubles to 23.7%[6]. This shows that the parser is often mistaking filler words as their non-filler counterpart homophones.

This combination of factors indicates there is now both incentive and enabling resources for a detailed study of how parsing and filler detection interact. Consequently, this proposal identifies two avenues for approaching this study: (1) establishing a common definition of fillers across SimpleMDE and tree annotation guidelines, and (2) evaluate how filler detection impacts parsing accuracy. In regard to (1), a significant hurdle to overcome is that SimpleMDE and treebanking guidelines do not share a consistent definition of fillers. Consequently, much of our preliminary work has involved studying annotation guidelines and existing annotated examples, working to understand the degree to which unification of these differing standards is possible and what revisions to current guidelines might be required for this resolution. While we have made some progress, more work remains to be done, and this topic is discussed at length in Section 2. In regard to (2), we propose an alternative syntactic representation of fillers that directly enables simple evaluation of how filler detection impacts

[2] http://www.darpa.mil/ipto/programs/ears

[3] While the Penn Treebank 3 (TB3) corpus also included filler markup as part of its disfluency annotation [6], SimpleMDE annotators listened to the audio whereas TB3 annotators did not [7]. SimpleMDE also represents a more recent and rigorous disfluency annotation standard than used in TB3, and joint use of TB3 syntax and disfluency annotations requires aligning the differing transcripts used in each annotation step (word and boundary variations), representing a potential source of additional error.

[4] See Section 2 for a description of how fillers are syntactically annotated.

[5] Error = the number of mislabeled words divided by the actual number of filler words.

[6] This result is for RT-04F Dev set #2 using reference words and SU boundaries.

parsing accuracy, as well as allowing filler detection and parsing models to be easily combined. We present this approach in detail in Section 3.

In addition to this focussed study on fillers, we also propose two other research tasks to address disfluency more broadly (as well as its interaction with parsing): (1) improving disfluency modeling by revising and extending existing noisy-channel based approaches, and (2) initiating cross-linguistic study of the interaction between parsing and disfluency modeling. Regarding (1), we believe existing approaches to disfluency modeling via parsing and the noisy-channel paradigm can be significantly refined and extended, with support added for incorporating new features such as prosody. These ideas are proposed in Section 4. Finally, we believe cross-linguistic study of the parsing-disfluency relationship is an important next step in demonstrating generality of existing disfluency modeling techniques and results. This is discussed further in Section 5.

2 Filler annotation consistency

In this section, we describe issues involved in achieving consistent annotation of fillers for data dually annotated under SimpleMDE and treebank guidelines [5, 9, 6, 10]. We believe such a consistent definition would be valuable in terms of establishing both standard vocabulary for the community and consistent resources for training and evaluating speech processing systems. We begin by describing how the corpus used in the JHU-SWLE-2005 Parsing And Structural Event Detection (PASSED) project was created, highlighting where consistency was and was not achieved. We then summarize existing SimpleMDE and treebanking guidelines specific to filler identification and annotation. Finally, we present a preliminary proposal for guideline revisions in order to achieve agreement between the two annotation standards.

2.1 Background

A prerequisite for the PASSED project was having a corpus dually annotated under SimpleMDE and treebanking guidelines. Because no such data existed prior to the workshop, LDC2005E15, *JHU Speech Parsing Workshop English TB Data*, was created by adding treebank annotations onto three corpora that had previously been SimpleMDE annotated for the Rich Transcription Fall 2004 (RT-04F) Evaluation: LDC2004E16, LDC2004E29, and LDC2004E50).[7]

The treebanking process began with automatic parsing of the RT-04F corpora using the Charniak parser [8]. In this stage, consistency between SimpleMDE and tree annotations was achieved as follows

1. **SUs**: annotated SU boundaries determined the extents of strings to be parsed and therefore the resulting trees

[7] The RT-04F training corpus, LDC2004E31, was unfortunately excluded from this effort due to budgetary constraints.

2. **edit words**: the parser was constrained such that any annotated edit word was required to descend from an EDITED node in the generated parse tree
3. **interruption-points (IPs)**: As is done in Switchboard [6], IPs were represented as (immobile) word tokens in the strings to be parsed
4. **filler words**: unlike the other types of metadata events, *no* effort was made at this stage to enforce consistency between annotated filler words and the resulting trees.

Following this automated stage, the trees generated by the parser were then handed off to LDC for correction, along with corresponding rich transcription strings[8] showing disfluency markup:

Baltimore is the greatest city in [America] * **uh i mean** the world

Treebanking was performed in accordance with existing treebank guidelines [9, 6] and additional guidelines developed during the treebanking process [10]. Because SimpleMDE annotators listened to the audio whereas the treebank annotators did not, SimpleMDE annotations were assumed to be correct by default and preserved as much as possible, as the guidelines indicate: *try to follow the disfluency annotation as much as possible, even if this sacrifices a more intelligible reading* [10]. Nevertheless, limited changes were made to annotated edit words and IPs in response to (1) obvious annotation error in the original SimpleMDE markup and (2) explicit differences between SimpleMDE and treebanking guidelines with regard to representation of complex edits[9]. In all such cases, changes were applied to both treebank and SimpleMDE markup to preserve consistency. Unfortunately, as was the case in the automatic parsing stage, effort was not made to preserve filler consistency between SimpleMDE and treebank annotations. Furthermore, preliminary inspection of the annotations produced suggests that such inconsistencies are not infrequent, with the consequence that the conflicting annotations suggest alternative readings of various utterances.

For both the automatic parsing and manual correction stages, it appears filler consistency was not specifically addressed due to lack of a clear understanding as to how annotated filler words should be represented in the corresponding trees. As such, we now turn our attention to summarizing how existing SimpleMDE and treebank guidelines identify and annotate fillers.

2.2 Fillers in SimpleMDE

SimpleMDE v6.2 [5] identifies four[10] types of fillers (emphasized text and examples below are excised from the annotation guidelines):

[8] The original RT-04F data had no punctuation and limited use of capitalization; punctuation was added by annotators during treebanking.

[9] While important, further discussion of this latter issue is unfortunately beyond the scope of this discussion.

[10] SimpleMDE also identifies Discourse Response (DR) as a subtype of Discourse Marker but states the DR concept is experimental and subject to revision. No DRs appear in the RT-04F corpora.

1. **Filled Pauses (FPs)**: *hesitation sounds that speakers employ to indicate uncertainty or to maintain control of a conversation while thinking of what to say next.* Examples include `ah, eh, er, uh, um`, etc.

2. **Discourse Markers (DMs)**: *words or phrases that function primarily as structuring units of spoken language.* Examples include `you know, i mean, like, so, well`, etc.

3. **Explicit Editing Terms (EETs)**: *overt statements from the speaker recognizing the existence of an edit disfluency.* Examples include `I mean, or rather`, etc.

4. **Asides and parentheticals (A/Ps)**: *asides occur when the speaker utters a short side comment on a new topic then returns to the main topic being discussed. Parentheticals are similar to asides in that they are brief remarks that break the flow of the larger utterance, but unlike asides the remark is on the same topic as the larger utterance.* Asides and parentheticals are not distinguished from one another in annotation. Examples include `oh I don't know, how long has it been, his words`, etc.

Annotation guidelines specify that words which can occur as fillers can generally also occur as non-fillers, hence annotators must be concious of surrounding context in determining whether or not a given word or phrase is acting as a filler. A paricular note is that words acting as backchannels, *words or phrases that provide feedback to the dominant speaker by indicating that the non-dominant speaker is still engaged in the conversation*, are not marked as fillers. Guidelines specify that in cases of uncertainty the word or phrase should not be annotated as a filler. Finally, in the event of multiple, contiguous fillers, annotators must decide if multiple words comprise a phrase filler or if each word is acting as an individual filler. Guidelines stipulate that *when in doubt, annotators should err on the side of separating the fillers into separate units.*

2.3 Fillers in tree annotation

In terms of tree annotation, fillers were first specifically addressed in Section 2.2 of the Switchboard addendum [6] to the original bracketing guidelines [9]. Emphasized text and examples below are excised from the addendum.

1. **Fillers**: *fillers are bracketed* INTJ. Examples include `uh, um`, etc.

2. **Discourse Markers**: *most discourse markers, like fillers, are labelled* INTJ. Examples include `well, like, now, see, say, actually, anyway`. *The discourse marker* `you know` *is labelled* PRN.

3. **Editing terms**: *Editing terms are labelled* PRN. Examples include `I mean, excuse me`, etc.

4. **Asides**: *Asides which interrupt the flow of a sentence are fully bracketed inside* PRN.

5. *Continuers and assessors of all types are labeled* INTJ *and constitute a sentence on their own.* Examples include `uh-huh, huh, really, exactly, right, yeah, oh, okay`, etc.

6. *Combinations of* INTJ*s like* `oh really, oh yeah`, *etc. are each labelled individually as* INTJ *and the highest label is also* INTJ.

In addition to the Switchboard addendum, an additional addendum [10] was created during treebanking of LDC2005E15 (the workshop corpus). Relevant details to mention include:

1. *all daughters of an INTJ node should have the UH POS tag*

2. *long renaming is enclosed in PRN*

For completeness, we note that the original treebank guidelines [9] and POS guidelines [11] make no explicit mention of fillers. The treebank guidelines state of the (INTJ) tag only that it *corresponds approximately to the part-of-speech (POS) tag* UH, and PRN is only loosely defined in terms of surrounding puncutation and annotator intuition, although the final example in Section 2 of the guidelines does show disfluent speech without any delimiting punctuation bracketed by PRN. The POS guidelines regarding UH are limited to one list of examples, without any definition of the term.

Finally, we note that the Switchboard corpus serves as a useful reference as to the resulting product of these rules. The top 97% of the INTJ word-phrase distribution and top 65% of the PRN distribution are conveniently listed by Engel et al. [1]. We assume phrasal INTJs shown in the listing there correspond to top-most INTJs, which, in turn, parent an additional INTJ for each of the terminals (per the last rule in the Switchbord addendum [6] cited above).

2.4 Unification of filler annotation schemes

After comparing SimpleMDE and treebank guidelines with regard to filler annotation, as well as preliminary inspection of the data to that end, we believe a consistent, unified treatment of filler annotation is possible. Our assumption for this unification, which is true at least of the workshop corpus, is that SimpleMDE annotation will be performed prior to treebanking, thus the primary goal is identifying how SimpleMDE filler annotations can be unambiguously mapped to tree annotation. This had led us to a preliminary proposal of a few additional rules for the treebanking guidelines

1. a word should descend from INTJ in the tree if and only if it is (a) marked as a complete filler in the SimpleMDE annotation or (b) part of a backchannel SU in the SimpleMDE annotation.

2. any phrase (group of multiple, contiguous words) marked as a single filler in the SimpleMDE annotation should descend from PRN in the corresponding tree

3. a short list of such known phrasal fillers, such as `you know` and `i mean`, etc., should descend from PRN only if they are marked as phrasal fillers in the SimpleMDE annotation.

The first rule is bidirectional and defines a mapping which conflates single word fillers and backchannel words in SimpleMDE as the tree INTJ category. The second rule is a unidirectional mapping from SimpleMDE phrasal fillers to the tree PRN category. The last rule is an admittedly simple attempt to catch annotation inconsistency of phrasal fillers marked in the tree and not in the SimpleMDE annotation. Following application of these rules, we assume the following property should hold

> for each instance of category INTJ and PRN, either all or none of its spanned terminals should be marked as fillers in corresponding SimpleMDE annotation

These rules (or a close approximation thereof) have been submitted to LDC and are pending review, but further examination of data and discussion with LDC is needed to determine if these particular rules are the right ones to adopt. One significant question is whether the PRN category and the class of phrasal fillers might, in fact, be equivalent. This would tremendously simplify the mapping between annotation schemes and definitions, but it is unclear whether or not this is reasonable. It could be that counterexamples exist but are rare, or that the counterexamples form a class that could represented in some other way. Another issue is that built into this proposal was a decision to maximize consistency with existing rules, thus the rules above preserve the status quo in bracketing both fillers and continuers/assessors as INTJ. Such overloading of INTJ is not necessarily a good idea, but changing this rule (e.g. via introduction of a new category) would require changing existing annotations accordingly. Finally, more data inspection in needed in verifying that backchannel words and tree *continuers and assessors* do indeed denote the same class of words.

A final note is merited in regard to the SimpleMDE annotation guideline that in cases of uncertainty, the word or phrase in question should not be annotated as a filler. As SimpleMDE annotations were specified in terms of transcript cleanup (i.e. marking fillers in transcripts to improve readability), this decision to not annotate a word or phrase as a filler in case of doubt may be viewed as deferring the decision to the eventual reader. Unfortunately, the situation is somewhat different when we consider SimpleMDE as an earlier annotation step to treebanking. Unlike SimpleMDE annotation, treebank annotation must commit to a specific analysis for the given utterance, hence a SimpleMDE deferral in case of uncertainty amounts to a final decision when SimpleMDE filler annotations are mapped to syntax as described above. As such, we also suggest that at least in the usage scenario we describe here, it would be best to encourage SimpleMDE annotators to take their best guess in cases of uncertainty rather than always leaving such cases unannotated.

3 A unified syntactic treatment of disfluency

In this section, we summarize the motivation for and details of a unified syntactic representation of disfluency. The starting point for this discussion is a revised syntactic representation for speech repairs proposed several years back [2] that has since been adopted by the JHU-SWLE-2005 Parsing And Structural Event Detection (PASSED) group and others [4]. This revised representation treats EDITED regions differently than other constituents due to their non-semantic nature, and we argue here that the arguments in favor of this representation apply equally well to fillers. Consequently, we believe there is value in extending this representational model of repairs to encapsulate disfluency in general, fillers as well as repairs.

3.1 Syntactic representation of speech repairs

Several years ago, Charniak and Johnson (C&J) argued that speech repairs should be afforded special treatment when parsing [2]. This argument rested on a fundamental assumption that speech repairs do not contribute toward the semantic and pragmatic content of an utterance, but rather could be considered largely as noise due to production error or other factors. To illustrate this point with a concrete example, consider the following two strings

> Baltimore is the greatest city in [America] * **uh i mean** the world

> Baltimore is the greatest city in the world

The claim of the assumption stated above is that these two strings are semantically and pragmatically equivalent: there is no loss in meaning by excising the repair. While the assumption does seem to hold for this example, C&J point out that counterexamples also exist:

> take the oranges to Elmira, um, I mean, take them to Corning

In this example (drawn from [12]), the antecedent of *them* is found in the repair, hence the repair does indeed contribute towards our understanding of the utterance. Nevertheless, C&J argue that examples like this are sufficiently infrequent to be of little practical consequence (relative to other errors made in automatic parsing).

The relevance of this assumption to parsing is that if one accepts repairs do not contribute toward semantic or pragmatic interpretation, C&J then question the value of

- recovering the internal structure of repairs
- distinguishing boundaries of multiple, contiguous repairs
- determining the correct attachment of repairs
- evaluating our ability to do any of the above

Instead, they argue our real concern should be simply discriminating between the meaningful and non-meaningful portions of utterances, evaluating our ability to (1) parse the former and (2) correctly identify boundaries of the latter. This stance has practical ramifications for parsing research, both in terms of parsing techniques and in how accuracy is measured. As parsing techniques have matured to the extent that reported improvements tend to be incremental rather that revolutionary these days, one could argue the community would be best served to focus future research on the meaning-bearing words to ensure we are not hill climbing techniques on semantically vacuous regions.

A related issue is determining how speech repairs can be best modeled. While one could simply let the parser detect these regions (i.e. EDITED constituents) along with other constituents, it is certainly conceivable than an alternative technique could be more effective, and this has, in fact, been shown, both in earlier study [4] and in the JHU-SWLE-2005 Parsing And Structural Event Detection (PASSED) results. Naturally, given a superior repair detection model, one would like to leverage it to improve parse accuracy, which presents a question of how to best integrate models. This is where another advantage of the C&J representation of repairs becomes apparent: such integration is trivial and independent of the models employed

1. detect the repairs with the repair model
2. parse the remaining words with the parser
3. reinsert each span of detected repair words under a flat EDITED constituent
4. relax PARSEVAL to ignore internal structure, contiguous boundaries, and attachment of EDITED constituents

Such separation of concerns between the parsing and repair detection models is valuable because each problem can be attacked independently, and it becomes trivial to measure how improvements in repair detection impact resulting parse accuracy. This was shown in in [2, 4] and in the PASSED results. Moreover, one can easily upperbound the improvement in parse accuracy possible through improved repair detection by assuming known repair boundaries (as an oracle experiment).

A general caveat of this approach worth noting, however, is that like any staged-system of hard decisions, later stages cannot inform or change decisions of earlier stages, and so errors tend to compound. If one were willing to accept a more complex model of integration between repair detection and parsing, an alternative model could have the repair detection module generate posteriors for its predictions (rather than hard decisions) which could then be combined with the parser's other knowledge sources in joint determination of repair boundaries.

3.2 Syntactic representation of fillers like speech repairs

In Section 2, we summarized current syntactic annotation guidelines [9, 6, 10] with respect to fillers. In particular, we observed that fillers appear to be bracketed with a combination of INTJ and PRN labels, and that these labels are also

used for non-filler constituents, such as backchannel feedback and long renaming. As this particular representation of fillers is but one of infinite possibilities, it is certainly possible an alternative representation might be easier to model or present some other practical benefit. For example, perhaps the filler and non-filler versions of INTJ and PRN have sufficiently different distributions that the current representation is mistakenly combining syntactically different phenomena. Removing such ambiguity could provide linguistic insights and improve automatic detection of correct constituency structure. Another example is that if one wishes to detect fillers in support of downstream processing, it is difficult to do this given the current syntactic representation due to the ambiguity just described. Finally, if we assume that like speech repairs, fillers do not contribute semantically or pragmatically towards the meaning of an utterance (and such an assumption seems inherent in any intuitive definition of the term *filler*), then we ought to treat fillers in much the same way repairs are treated. In particular, if one accepts the assumptions and motivations of the syntactic representation of repairs proposed in [2], then it seems the same arguments apply equally well to fillers.

As such, we propose the following new syntactic representation of disfluency, intended to extend the C&J representational model of repairs to encapsulate disfluency in general, fillers and repairs. We assume here that prior to performing this transformation, SimpleMDE and tree annotations will have been made consistent per the rules specified in Section 2. Specifically, we assume that for each instance of category INTJ and PRN, either all or none of its spanned terminals should be marked as fillers in corresponding SimpleMDE annotation. If all of the category's terminals are marked as fillers, we define it to be a *filler-type* instance of the given category.

1. relabel as FILLER any filler-type (non-backchannel) instance of INTJ
2. relabel as FILLER any filler-type instance of PRN
3. relax PARSEVAL to ignore internal structure, contiguous boundaries, and attachment of EDITED and FILLER constituents. The distinction between adjacent EDITED and FILLER will be evaluated, but FILLER constituents within EDITED constituents will be discarded along with other internal structure.

The first two steps (1) remove the ambiguity of INTJ and PRN labels by using them to represent non-fillers only, and (2) collapse the distinction between INTJ and PRN fillers by representing both as FILLER constituents. The last step relaxes filler evaluation similar to C&J's treatment of EDITED [2]. Note that fillers can occur within repairs but not vice-versa, so if a given word is detected to be both an edit and a filler word, it should be labelled in the generated tree as a FILLER below an EDITED in order to match the gold tree in evaluation.

It is worth discussing here why we do not collapse EDITED and FILLER into a single DISFLUENT (or perhaps NON-SEMANTIC) category. While we have argued that neither repairs nor fillers contribute toward semantic or pragmatic

interpretation, they do significantly differ from one another distributionally and structurally (e.g. repairs have crossing dependencies and fillers do not, as discussed in Section 4.2). Besides suggesting that distinct cognitive phenomena are at work, such differences will also likely result in varying detection accuracies in the two cases. Consequently, we would like to be able to separately measure the impact of each form of disfluency on parse accuracy, and representing this distinction syntactically makes it relatively easy to perform such evaluation. Finally, in terms of downstream tasks, it is certainly conceivable that applications besides Rich Transcription might also find it useful to differentiate repairs from fillers, and such differentiation ought to be possible from the syntax.

To summarize, the proposed representation offers the following benefits

- fillers can be distinguished from non-fillers using syntax
- the impact of fillers on parse accuracy can be easily measured
- we can easily plug-in alternative models of filler detection
- we focus parser evaluation on the meaningful words and avoid the possibility of hill-climbing parsing techniques on meaningless words

4 Improving noisy-channel based models of disfluency

In this section, we propose improving disfluency modeling by refining and extending noisy-channel based techniques that have been reported in the literature. As use of the noisy-channel paradigm for disfluency modeling was first proposed by Honal and Schultz [13, 14], we begin by summarizing their work. Following this, we then describe an alternative model put forward by Johnson and Charniak [3] based on the *tree adjoining grammar* (TAG) linguistic formalism. After summarizing their work, we propose several ways in which we believe the existing model can be significantly improved. Finally, we present Knight and Marcu's noisy-channel model for text compression [15] and describe how it could be used to recognize regularities of filler syntax and parser errors arising from mistakes in filler analysis.

4.1 Noisy-channel modelling of disfluency

The idea of applying the noisy-channel paradigm to disfluency modeling was first proposed by Honal and Schultz (H&S) [13, 14]. To describe how this model operates, we begin by imagining that speakers intend to say fluent (*clean*) utterances C but actually produce observed utterances N that may be *noisy* due to optional disfluency insertion[11]. For each observation, the goal is to recover the

[11] Use of the noisy-channel paradigm for disfluency modeling is simpler than in the case of statistical machine translation in that it is assumed noise arises only from insertion errors and that source words are not deleted or re-ordered.

most likely underlying source utterance $\widehat{c}$. Applying Bayes Rule, this problem can be formulated in canonical noisy channel form

$$\widehat{c} \quad = \quad \underset{C}{\operatorname{argmax}} \, \mathrm{P}(C|N) = \underset{C}{\operatorname{argmax}} \, \mathrm{P}(N|C)\mathrm{P}(C)$$

where $\mathrm{P}(N|C)$, the *channel model*, defines a stochastic mapping of source utterances into observations via the optional insertion of disfluency, and $\mathrm{P}(C)$, the *language model*, defines a probability distribution over source utterances.

In H&S's implementation, $\mathrm{P}(C)$ is realized as an n-gram language model trained on fluent speech. The channel model, which generates N given C in left-to-right fashion, is defined as follows

- let J be the length and n_j the words of the source sentence N[12]
- let I be the length and c_i the words of the target sentence C
- let m denote the number of deletions (of contiguous word sequences) made during generation of the target sentence
- let a_j define whether the word n_j in the source sentence is deleted or preserved in the target sentence

Given this, the channel probability is then defined as

$$P(N|C) = P(J|I) \cdot P(m|J,C) \cdot \sum_{a_1^J} P(n_1^J, a_1^J | c_1^I, I, J, m)$$

where the probability $P(n_1^J, a_1^J | c_1^I, I, J, m)$ can be decomposed into a product of probabilities over all source words n_j. The probability assigned to each word is calculated as a weighted sum of five models that each consider various features, such as disfluency length and position, etc.

While H&S's overall model is compelling and the results are impressive, we believe there are several ways in which their formulation might be improved upon. For example, one might use a syntax-based language model (as done in [3]) rather than an n-gram model. It also seems significant improvement could be had through additional feature engineering, such as distinguishing between repairs and fillers and leveraging acoustic information. Besides these minor points, a more serious criticism must also be levelled at the experimental setup employed, in which the English filler annotations used for training and evaluation were generated automatically[13]. The unfortunate consequence of this setup is that it remains unclear how well the system would perform on manually annotated data. A final note is that examination of the system probability models suggests that fluent words are being generated by the channel model as well as the language model, making the overall probability model deficient. While

[12] Our presentation follows Honal and Schultz's in identifying the generated sentence N to be the source sentence and the input C as the target.

[13] The system used is reported to achieve 51.9% recall and 59.4% precision [13].

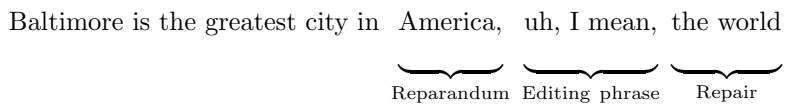

Figure 1: The structure of a typical repair.

it would certainly be possible to reformulate the model in response to this and the other issues mentioned above, we believe a more fruitful starting point for future work in noisy-channel based disfluency modeling is to build upon one or more of the alternative model formulations described in the subsequent sections.

4.2 A TAG-based noisy-channel model of disfluency

Johnson and Charniak (J&C) have recently proposed a noisy-channel model of repairs based on the *tree adjoining grammar* (TAG) linguistic formalism [3]. Following Heeman [16], J&C observed that in the structure of a typical repair (an example is shown in Figure 1), the correction is often a *rough-copy* of the reparandum, using the same or very similar words in roughly the same order. In other words, a speech repair seems to involve *crossing dependencies* between the reparandum and the correction. Unfortunately, HMM and PCFG models[14] can only induce linear or tree-structured dependencies between words, meaning they cannot capture the type of depdencies found in repairs.

Unlike HMM and PCFG models, TAGs can model such dependencies. TAGs provide a systematic way of formalizing the channel model, and their polynomial-time dynamic programming parsing algorithms can be used to search for likely repairs, at least when used with simple language models like a bigram language model. Consequently, J&C first identify a candidate set of most likely analyses of each sentence using the TAG channel model together with a bigram language model. Following this, each of these analyses is rescored using the TAG channel model and a syntactic parser based language model [17], which revises the bigram model's likelihood estimate of each fluent candidate.

There are several ways to define transducers using TAGs, such as [18], and J&C adopt a simple method inspired by finite-state transducers. The TAG defines a language whose vocabulary is the set of pairs $(\Sigma \cup \{\emptyset\}) \times (\Sigma \cup \{\emptyset\})$, where Σ is the vocabulary of the observed sentences Y. A string Z in this language can be interpreted as a pair of strings (Y, X), where Y is the concatenation of the projection of the first components of Z and X is the concatentation of the projection of the second components. For example, the string Z = *a:a flight:flight to:∅ Boston:∅ uh:∅ I:∅ mean:∅ to:to Denver:Denver on:on Friday:Friday* corresponds to the observed string Y = *a flight to Boston uh I mean to Denver on Friday* and the source string X = *a flight to Denver on Friday.*

[14] An example PCFG is the Charniak parser [8].

Nonterminals in the TAG grammar are of the form N_{w_x}, $R_{w_y:w_x}$ and I, where w_x is a word appearing in the source string and w_y is a word appearing in the observed string. Informally, the N_{w_x} nonterminals indicate that the preceding word w_x was analysed as not being part of a repair, while the $R_{w_y:w_x}$ that the preceding words w_y and w_x were part of a repair. The nonterminal I generates words in the interregnum (editing phrase) region of a repair. Encoding the preceding words in the TAGs nonterminals permits the channel model to be sensitive to lexical properties of the preceding words. The start symbol is $N_{\$}$, where '$' is a distinguished symbol used to indicate the beginning and end of sentences.

To estimate weights on the TAG productions one needs to know the TAG derivation of each sentence in the training data. In order to uniquely determine this one needs to know the not just the locations of each reparandum, interregnum, and correction, but also the crossing dependencies between the reparandum and correction words. J&C obtain these by aligning the reparandum and repair strings of each repair using a minimum-edit distance string aligner. Alignment costs are chosen so that a substitution will be selected over an insertion followed by a deletion, and the lower cost for substitutions involving POS tags beginning with the same letter is a rough and easy way of establishing a preference for aligning words whose POS tags come from the same broad class, e.g., it results in aligning singular and plural nouns, present and past participles, etc.

From the training data, a number of conditional probability distributions are estimated. These distributions are the linear interpolation of the corresponding empirical distributions using various subsets of conditioning variables (e.g., bigram models are mixed with unigram models, etc.) using Chen's bucketing scheme [19]. As is commonly done in language modelling, the interpolation coefficients were determined by maximizing the likelihood of the held out data counts using EM. Special care was taken to ensure that all distributions over words ranged over (and assigned non-zero probability to) every word that occurred in the training corpora.

Given this summary of the C&J model, we now discuss our proposed work in refining and extending it. The most significant limitation of the existing formulation is that it does not model filler disfluencies except for those appearing in the context of repairs. As such, the grammar and relevant features need to be expanded in order to realize a more general model of disfluency. Another significant limitation of the current model is its dependence on training data that has the correction annotated, as described above. While such annotations were provided in Switchboard, SimpleMDE [5] does not annotate the correction portion of repairs, and it is under this annotation standard that Fisher and other conversational speech transcripts will be annotated for the forseeable future. In the RT-04F evaluation, C&J worked around this limitation by training their model on Switchboard while evaluating it on the Fisher test data [20]. However, different annotation guidelines regarding sentence-like unit (SU) boundaries and repairs cause this approach to suffer from a train-test mismatch. To remedy this, we propose using an EM-style approach to automatically annotate the correction

term. In this approach, the channel is constrained to generate the annotated repair and editing phrase, allowing it to effectively re-estimate the location of the correction term after iteration. In addition to addressing the train-test mismatch issue, it will also be generally valuable to be able to automatically determine the correction region of a repair given only the reparandum and editing phrase. A final issue we would like to explore is incorporating acoustic-based features into the model. While such features could be easily incorporated into the reranking phase of the model [20], more sophistication integration of such features into the TAG itself could serve to improve quality of the first and second stage candidate sets fed to the reranker, thereby improving the upperbound performance of the overall model.

4.3 Finding disfluency-detection error patterns via text-compression

In this section, we summarize Knight and Marcu's (K&M) noisy-channel model for text compression [15], describing how it could be used to recognize syntactic filler patterns and regularities of parser errors arising from mistakes in filler analysis. *Text compression* as defined by K&M involves determining how long sentences can be automatically shortened with minimal impact on meaning and grammaticality. They approach this problem via the noisy-channel paradigm by assuming the longer version of a sentence is a noisy version of its shorter counterpart. Key to the discussion here is that the only noisy operation allowed in their channel model is the insertion of noisy words. We observe that this setup parallels that of disfluency modeling, in which the only noisy operation permitted is the optional insertion of disfluent words into otherwise fluent utterances. Because of this, it seems very natural to consider whether the K&M's model formulation also be applied to disfluency modeling.

In the K&M model, $P(s)$ is the source model: the probability that s is the original sentence. $P(l \mid s)$ is the channel model: the probability that when the short sentence is expanded, the long sentence is the result. Instead of working solely with strings, the K&M model uses parse trees for the sentences. These allow it both to better determine the probability of the short sentence and to obtain alignments from the training data. In the original model the sentence probability is determined by combining a probabilistic context free grammar with a word-bigram score. The joint rules used in creating the compressions are generated by aligning the nodes of the short and long trees in the training data in order to determine expansion probabilities $P(l \mid s)$.

Recall that this language model is attempting to find the probability of the long string with respect to the short string. It obtains these probabilities by aligning nodes in the parallel training corpus, and counting the nodes that align as "joint events." For example, there might be an instance of S $\rightarrow$ NP VP PP in a long sentence and S $\rightarrow$ NP VP as the short sentence, and one joint event is counted for this occurrence. Non-compressions, where the long version is the same as the short version, are also counted. The expansion probability is given

by

$$\frac{P_{expand}(l|s) = count(joint(l,s))}{count(s)} \quad (1)$$

where $count(joint(l,s))$ is the number of times the long rule matches up with the short rule. Many compressions do not align exactly. Sometimes the parses do not match, and sometimes there are deletions that are too structurally complex to be modeled in this way. In these cases sentence pairs, or sections of them, are ignored.

The original model is implemented by creating a packed parse forest of all possible compressions that are grammatical with respect to the Penn Tree-bank. Any compression given a zero expansion probability according to the training data is instead assigned a very small probability. A tree extractor [21] is used to collect the short sentences with the highest score for $P_{expand}(s \mid l)$, based on both the compression probabilities and the word-bigram probabilities. While the program may return a list of trees (one for each possible compression length) a best compression is obtained by weighting the probability assigned by the model vs sentence length.

In terms of disfluency modeling, our general goal is to use the K&M approach to model syntactic regularities regarding disfluencies so that they may be *compressed* out (i.e. detected). First of all, we note that this approach will have no value in identifying repairs in human-annotated gold trees because the syntactic pattern of them is trivial: all terminals under an EDITED node comprise a repair and so may be compressed out. However, it may be that this approach could be used to learn syntactic regularies in cases where the generated parse did not identify EDITED nodes correctly. That is, we would hope to detect syntactic regularities between the true parse of the short, fluent sentence and the generated parse of the long, noisy sentence in cases of such errors.

In terms of fillers, it should be recalled that it is currently unclear to what extent INTJ and PRN labels denote fillers and not other phenomena as well (Section 2). If it were the case that there was a 1:1 mapping between these labels and fillers, then as with repairs, there would be no need for automatic pattern detection. However, if it should turn out to be unclear exactly how filler words align to these labels, then K&M's method could prove useful in determining syntactic patterns between fluent utterances and their noisy counterparts. Either way, we also hope that as with repairs, K&M's method could prove useful in identifying syntactic regularities of mistakes made by the parser in misapplication of INTJ and PRN labels with respect to true fillers.

This portion of the proposed research will build on existing work with the K&M model performed by Jenine Turner and Eugene Charniak [22].

5 Cross-linguistic study of disfluency and parsing

Little work has been done to date in evaluating the relationship between disfluency detection and parse accuracy cross-linguistically. One significant reason for this is the limited availability of data annotated for such analysis. However, as part of JHU-SWLE-2005, a portion of the Levantine CallHome corpus of transcribed conversational speech was treebanked for use in the *Parsing Arabic Dialects* (PAD) project. Due to the complexity of their task, the PAD group chose to filter all disfluencies from their test data as a simplifying assumption, and reported a resulting 21% improvement in parsing f-measure[15]. While this result should be treated as preliminary, it is likely indicative of a cross-linguistic finding that standard parsing techniques in use today are less effective in the presence of disfluency, or said another way, do not model disfluent speech as well as fluent speech. This result in Levantine is of further interest because LDC recently released a small portion of the Levantine CallHome transcripts with SimpleMDE annotations, LDC2004E47 (a pilot release), so we now have an opportunity in Levantine as well as English to begin looking at how SimpleMDE and tree annotations interact. While a similar pilot SimpleMDE corpus of annotated Mandarin CallHome transcripts as also been released (same catalog ID), no treebanking of the Mandarin CallHome transcripts has yet been performed. Nevertheless, current trends in languages of interest to funding agencies suggest such data may also become available in the near future, which would provide another language for comparison. In all such cases, it will be interesting to begin porting disfluency modeling techniques shown to be effective in English to these other languages and measuring how well they perform and the relative impact on parsing accuracy (as well as exploring the other direction of how parsing can improve disfluency detection). This work may highlight issues in the disfluency-parsing interaction not yet encountered in English, and suggest new avenues for research in these related fields.

6 Conclusion

In this paper, we have outlined four areas for future research: (1) establishing a common definition of fillers across SimpleMDE and tree annotation guidelines, (2) investigating an alternative syntactic representation of fillers for more accurate parsing and more meaningful parser evaluation, (3) improving disfluency modeling by revising and extending existing noisy-channel based approaches, and (4) initiating cross-linguistic study of the interaction between parsing and disfluency modeling. This work will provide a valuable extension to the Parsing And Structural Event Detection (PASSED) research performed at JHU-SWLE-2005 by continuing to further our understanding of how parsing and disfluency models interact, particularly as we work toward increasingly automated systems

[15] Details of this experiment can be found in the PAD final report.

for rich transcription and analysis of conversational speech.

References

[1] Engel, D., Charniak, E., Johnson, M.: Parsing and disfluency placement. In: Proc. EMNLP. (2002) 49–54

[2] Charniak, E., Johnson, M.: Edit detection and parsing for transcribed speech. In: Proc. NAACL. (2001) 118–126

[3] Johnson, M., Charniak, E.: A TAG-based noisy channel model of speech repairs. In: Proc. ACL. (2004) 33–39

[4] Kahn, J.G., Lease, M., Charniak, E., Johnson, M., Ostendorf, M.: Effective use of prosody in parsing conversational speech. In: Proc. of EMNLP-05. (2005)

[5] LDC: Simple Metadata Annotation Specification Version 6.2. (2004) Available at http://www.ldc.upenn.edu/Projects/MDE.

[6] Taylor, A.: Bracketing Switchboard: An addendum to the Treebank II Bracketing Guidelines. Linguistic Data Consortium. (1996)

[7] Graff, D., Bird, S.: Many uses, many annotations for large speech corpora: Switchboard and TDT as case studies. In: Proc. LREC. (2000) 427–433

[8] Charniak, E.: A maximum-entropy-inspired parser. In: Proc. NAACL. (2000) 132–139

[9] Bies, A., Ferguson, M., Katz, K., MacIntyre, R.: Bracketting Guideliness for Treebank II style Penn Treebank Project. Linguistic Data Consortium. (1995)

[10] Bies, A., Mott, J., Warner, C.: Addendum to the Switchboard Treebank Guidelines. Linguistic Data Consortium. (2005)

[11] Santorini, B.: Part-of-speech tagging guidlines for the penn treebank project (3rd revision). Technical report, University of Pennsylvania (1990) Available as http://www.cis.upenn.edu/ treebank/home.html.

[12] Core, M., Schubert, L.: A syntactic framework for speech repairs and other disruptions. In: Proc. ACL. (1999) 413–420

[13] Honal, M., Schultz, T.: Correction of disfluencies in spontaneous speech using a noisy-channel approach. In: Proc. of Eurospeech-03. (2003)

[14] Honal, M., Schultz, T.: Automatic disfluency removal on recognized spontaneous speech – rapid adaptation to speaker-dependent disfluencies. In: Proc. of ICASSP-05. (2005)

[15] Knight, K., Marcu, D.: Statistics-based summarization - step one: Sentence compression. In: Proc. of AAAI-00. (2000)

[16] Heeman, P.A., Allen, J.F.: Speech repairs, intonational phrases, and discourse markers: Modeling speaker's utterances in spoken dialogue. Computational Linguistics **25** (1999) 527–571

[17] Charniak, E.: Immediate-head parsing for language models. In: Proc. ACL. (2001)

[18] Shieber, S.M., Schabes, Y.: Synchronous tree-adjoining grammars. In: Proceedings of the 13th International Conference on Computational Linguistics (COLING 1990). (1990) 253–258

[19] Chen, S.F., Goodman, J.: An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University (1998)

[20] Johnson, M., Charniak, E., Lease, M.: An improved model for recognizing disfluencies in conversational speech. In: Proc. of the Rich Text 2004 Fall Workshop. (2004)

[21] Langkilde, I.: Forest-based statistical sentence generation. In: Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computationl Linguistics. (2000)

[22] Turner, J., Charniak, E.: Supervised and unsupervised learning for sentence compression. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), Ann Arbor, Michigan, Association for Computational Linguistics (2005) 290–297