Conversation Trees: A Grammar Model for Topic Structure in Forums

Annie Louis and Shay B. Cohen
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
{alouis,scohen}@inf.ed.ac.uk

Abstract

Online forum discussions proceed differently from face-to-face conversations and any single thread on an online forum contains posts on different subtopics. This work aims to characterize the content of a forum thread as a *conversation tree* of topics. We present models that jointly perform two tasks: segment a thread into subparts, and assign a topic to each part. Our core idea is a definition of topic structure using probabilistic grammars. By leveraging the flexibility of two grammar formalisms, Context-Free Grammars and Linear Context-Free Rewriting Systems, our models create desirable structures for forum threads: our topic segmentation is hierarchical, links non-adjacent segments on the same topic, and jointly labels the topic during segmentation. We show that our models outperform a number of tree generation baselines.

1 Introduction

Online forums are commonplace today and used for various purposes: product support and troubleshooting, opining about events and people, and student interaction on online course platforms. Threads in these forums become long, involve posts from multiple users, and the chronological order of the posts in a thread does not represent a continuous flow of dialog. Adding structure to these threads is important for tasks such as information extraction, search, and summarization.

One such aspect of structure is *topic*. Figure 1 shows a computer-troubleshooting related thread with six posts. The first post is the troubleshooting question and the remaining posts can be seen as focusing on either of two topics, the *driver software* (posts p_1, p_2, p_5) or the *speaker hardware* (p_3, p_4). By categorizing posts into such topics, we can provide a useful division of content in a thread and even across multiple threads. Note that the *driver* topic is not a contiguous sequence but present in non-adjacent parts, (p_1, p_2) and (p_5).

p_0 Bob:	When I play a recorded video on my camera, it looks and sounds fine. On my computer, it plays at a really fast rate and sounds like Alvin and the Chipmunks!
p_1 Kate:	I'd find and install the latest audio driver.
p_2 Mary:	The motherboard supplies the clocks for audio feedback. So update the audio and motherboard drivers.
p_3 Chris:	Another fine mess in audio is volume and speaker settings. You checked these?
p_4 Jane:	Yes, under speaker settings, look for hardware acceleration. Turning it off worked for me.
p_5 Matt:	Audio drivers are at this link. Rather than just audio drivers, I would also just do all drivers.

Table 1: Example forum thread conversation

We tackle the problem of joint topic segmentation and topic labeling of forum threads. Given a thread's posts in chronological order (the order in which they were posted), we create a phrase structure tree indicating how the posts are grouped hierarchically into subtopics and super-topics. In these *conversation trees*, leaves span entire posts. Each non-terminal identifies the topic characterizing the posts in its span. Topics are concepts or themes which summarize the content of a group of posts. Specifically, a topic is a set of words which frequently co-occur in posts which are similar in content and other conversation regularities.

Our key insight in this work is to formalize topic structure using probabilistic grammars. We define a base grammar for topic structure of forum threads and refine it to represent finer topics and subtrees. We learn to predict trees under our grammar based on two formalisms: Probabilistic Context-Free Grammars (PCFG) and Probabilistic Linear Context-Free Rewriting Systems (PLCFRS). In the PCFG model, a non-terminal spans a contiguous sequence of posts. In the

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1566–1576,
Lisbon, Portugal, 17-21 September 2015. ©2015 Association for Computational Linguistics.

PLCFRS model, non-terminals are allowed to span discontinuous segments of posts. We leverage algorithms from probabilistic parsing of natural language sentences and modify them for our domain. We show that our model performs well and sidesteps a number of limitations of prior topic segmentation approaches. In particular:

- Our models perform joint topic segmentation and topic labeling while most existing models identify unlabeled segments. Labeling topics on segments creates richer annotation, and links non-adjacent segments on the same topic.
- Our grammar-based probabilistic models have two key benefits. They naturally create tree structures which are considered linguistically suitable for topic segmentation but were difficult to create under previous approaches. Second, the flexibility of grammars such as PLCFRS allow our models to seamlessly learn to produce trees where non-adjacent segments on the same topic are explicitly linked, an issue that was not addressed before.

We present large-scale experiments on a collection of forum threads from the computer-troubleshooting domain.[1] We show that our grammar models achieve a good balance between identifying when posts should be in the same topic versus a different topic. These grammar models outperform other tree generation baselines by a significant margin especially on short threads.

2 Related work

The ideas in this paper are related to three areas of prior research.

Forum thread analysis. Finding structure in forum threads has been previously addressed in two ways. The first is reply structure prediction where a parent post is linked to its children (replies) which were posted later in time (Wang et al., 2008; Cong et al., 2008). Reply links are sometimes augmented with a dialog act label indicating whether the child post is a question, answer, or confirmation to the parent post (Kim et al., 2010; Wang et al., 2011). The second set of methods partition sentences in emails or blog comments into topical clusters and then show salient words per cluster as topic tags (Joty et al., 2013).

We focus on producing rich hierarchical segmentation going beyond clusters which do not contain any cluster-internal structure. We also believe that topic structure is complementary to dialog act and reply link annotations. Tasks on forum data such as user expertise (Lui and Baldwin, 2009) and post quality prediction (Agichtein et al., 2008), and automatic summarization (Nenkova and Bagga, 2003) can be carried out on a fine-grained level using topic information.

Conversation disentanglement. A related problem of clustering utterances is defined specifically for Internet Relay Chat (IRC) and speech conversations. In this case, multiple conversations, on different topics, are mixed in and systems extract threads which separate out individual conversations (Shen et al., 2006; Adams and Martell, 2008; Elsner and Charniak, 2010).

Disentanglement is typically applied for the coarse-level problem of identifying a coherent conversation. In addition, these methods do not create any structure upon the clustered utterances in contrast to the focus of this work.

Topic Segmentation. is a task which directly focuses on the topic aspect of text and speech. This task is of greater importance for speech which lacks explicit structure such as paragraphs and sections. Many approaches perform linear segmentation where boundaries are inserted in the text or speech to divide it into a flat set of topic segments (Hearst, 1994; Utiyama and Isahara, 2001; Galley et al., 2003; Malioutov and Barzilay, 2006; Eisenstein and Barzilay, 2008). Very few methods recursively combine smaller segments into larger ones. Such hierarchical models (Eisenstein, 2009) have been applied at a coarse level for segmenting very long texts such as books into sections. Other work has focused on linear segmentation for documents within the same domain and having a regular structure (Chen et al., 2009; Jeong and Titov, 2010; Du et al., 2015). These latter approaches rely on three assumptions: that the documents contain a regular set of topics, that these topics are discussed in a fairly regular consensus order, and that the same topic does not recur in the same document.

Our models address two deficiencies in these approaches. First, text is commonly understood to have a hierarchical structure (Grosz and Sidner, 1986) and our grammar model is an ideal framework for this goal. Tree structures also have other advantages, for example, we do not predefine the number of expected topic segments in a conversation tree, a requirement posed by many prior seg-

[1] Our corpus is available from `http://kinloch.inf.ed.ac.uk/public/CTREES/ConversationTrees.html`

mentation algorithms. The second limitation of prior studies is assuming that topics do not recur in the same document. But linguistic theories allow for non-adjacent utterances to belong to the same topic segment (Grosz and Sidner, 1986) and this fact is empirically true in chat and forum conversations (Elsner and Charniak, 2010; Wang et al., 2011). Our models can flexibly handle and link recurring topics within and across threads.

As a final note, because of the annotations required, most prior work on forums or IRC chats have typically used few hundred threads. We present a heuristically derived large corpus of topic structure on which we evaluate our models.

3 Background

Our topic discovery methods are based on two constituency grammar formalisms.

3.1 Probabilistic Context-Free Grammars

A PCFG is defined by a 5-tuple G_C =(N, T, P, S, D) where N is a set of non-terminal symbols, T a set of terminal symbols, and S is the start symbol. P is a set of production rules of the form $A \rightarrow \beta$ where A is a non-terminal and $\beta \in \{N \cup T\}^*$. D is a function that associates each production rule with a conditional probability of the form $p(A \rightarrow \beta|A)$. This probability indicates how often the non-terminal A expands into β. The probabilities of all the rules conditioned on a particular non-terminal should sum to 1.

The joint probability of a tree T with yield Y, $P(T, Y)$, is the product of the probabilities of all the productions used to construct T. The parsing problem is to find the tree $\hat{T}$ which is most likely given the yield Y. $\hat{T} = \arg\max_T P(T|Y) = \arg\max_T P(T, Y)$.

Given training trees, we can enumerate the productions and compute their probabilities using maximum likelihood estimates (MLE):

$$p(A \rightarrow \beta|A) = \frac{\text{count}(A \rightarrow \beta)}{\text{count}(A)}$$

which is the fraction of the times the non-terminal A expands into β.

The most likely parse tree can be found using a number of algorithms. In this work, we use the CYK algorithm for PCFGs in Chomsky Normal Form. This algorithm has complexity $\mathcal{O}(n^3)$ where n is the length of the yield.

PCFGs do not capture a frequently occurring property of forum threads, discontinuous segments on the same topic. Indirectly however, a PCFG may assign the same non-terminal for each of these segments. To model these discontinuities more directly, we present a second model based on PLCFRS where non-terminals are allowed to span discontinuous yield strings.

3.2 Probabilistic Linear Context-Free Rewriting Systems

LCFRS grammars (Vijay-Shanker et al., 1987) generalize CFGs, where non-terminals can span discontinuous constituents. Formally, the span of an LCFRS non-terminal is a tuple, with size $k \geq 1$, of strings, where k is the non-terminal "fan-out". As such, the fan-out of a CFG non-terminal is 1.

An LCFRS G_L =(N, T, P, S, V) where N is the set of non-terminals, T the terminals and S is the start symbol. A function $f : N \rightarrow \mathbb{N}$ gives the fan-out of each non-terminal. P is the set of productions or otherwise called rewriting rules of the LCFRS. V is a set of variables used to indicate the spans of each non-terminal in these rules. A rewriting rule has the form:

$$A(\alpha_1, \alpha_2, \ldots, \alpha_{f(A)}) \rightarrow \\ A_1(x_1^1 \ldots, x_{f(A_1)}^1), \ldots, A_m(x_1^m, \ldots, x_{f(A_m)}^m)$$

Here A,A_1,. . .,$A_m \in N$. Since there are m non-terminals on the RHS, this rule has rank m. $x_j^i \in V$ for $1 \leq i \leq m$ and $1 \leq j \leq f(A_i)$ indicate the $f(A_i)$ discontinuous spans dominated by A_i. $\alpha_i \in (T \cup V)^*$, $1 \leq i \leq f(A)$ are the spans of the LHS non-terminal A.

A rewriting rule explains how the left-hand side (LHS) non-terminal's span can be composed from the yields of the right-hand side (RHS) non-terminals. For example, in the rule $A(x_1x_2, x_3) \rightarrow B(x_1)C(x_2, x_3)$, A and C have fan-out 2, B has fan-out 1. The two spans of A, x_1x_2 and x_3, are composed from the spans of B and C. For comparison, the productions of a CFG take the single spans of each non-terminal on the RHS and concatenate them in the same order to yield a single span of the LHS non-terminal.

A Probabilistic LCFRS (PLCFRS) (Levy, 2005) also contains D, a function which assigns conditional probabilities p($A(\vec{x}) \rightarrow \vec{\phi}|A(\vec{x})$) to the rules. The probabilities conditioned on a particular non-terminal and span configuration, $A(\vec{x})$ should sum to 1. Given a training corpus, LCFRS rules can be read out and probabilities computed similar to CFG rules.

To find the most likely parse tree, we use the parsing algorithm proposed by Kallmeyer and Maier (2013) for binary PLCFRS. The approach uses weighted deduction rules (Shieber et al., 1995; Nederhof, 2003), which specify how to compute a new item from other existing items. Each item is of the form $[A, \vec{\rho}]$ where A is a non-terminal and $\vec{\rho}$ is a vector indicating the spans dominated by A. A weight w is attached to each item which gives the $|\log|$ of the Viterbi inside probability of the subtree under that item. A set of goal items specify the form of complete parse trees. By using the Knuth's generalization (Knuth, 1977) of the shortest paths algorithm, the most likely tree can be found without exhaustive parsing as in Viterbi parsing of CFGs. The complexity of parsing is $\mathcal{O}(n^{3k})$ where k is the fan-out of the grammar (the maximum fan-out of its rules).

4 Problem Formulation

Given a thread consisting of a sequence of posts (p_1, p_2, …, p_n) in chronological order, the task is to produce a constituency tree with yield (p_1, p_2 …p_n). A leaf in this tree spans an entire post. Non-terminals identify the topic of the posts within their span. Non-terminals at higher levels of the tree represent coarser topics in the conversation (the span covered by these nodes contain more posts) than those lower in the tree. The root topic node indicates the overall topic of the thread.

Below we define a Context-Free Grammar (CFG) for such trees.

4.1 A Grammar for Conversation Trees

G_B is our *base grammar* which is context-free and has four non-terminals $\{S, X, T, C\}$. Each post p in the corpus is a terminal symbol (i.e. a terminal symbol is a bag of words). The productions in G_B are: $S \rightarrow T\ X^*$, $X \rightarrow T\ X^*$ and $T \rightarrow p$.

G_B generates trees with the following structure. A root-level topic S characterizes the content of the entire thread. *Thread-starting* rules are of the form, $S \rightarrow T\ X^*$, where X^* indicates a sequence of zero or more X non-terminals. T nodes are pre-terminals analogous to part-of-speech tags in the case of syntactic parsing. In our grammar, the $T \rightarrow p$ rule generates a post in the thread. In the *thread-starting* rules, T generates the first post of the thread which poses the query or comment that elicits the rest of the conversation. The X^* sequence denotes *topic branches*, the subtree under each X is assumed to correspond to a different topic. These X's characterize all but the first post of the thread. The *continuation rules*, $X \rightarrow T\ X^*$, recursively subdivide the X subtrees into topics spanning fewer posts. In each case, the T node on the right-hand side of these rules generates the first post (in terms of posting time) in that subtree. Therefore posts made earlier in time always dominate (in the tree structure) those which come later in the thread. We define the head of a non-terminal as the first post as per the chronological order of posts in the span of the non-terminal.

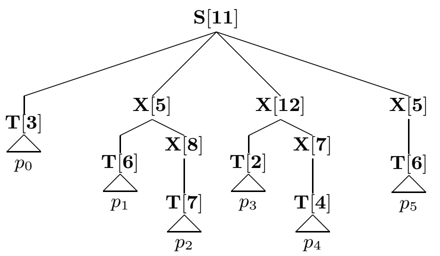


Figure 1: Example conversation tree for the thread in Table 1

This grammar does not generate binary trees. We binarize the tree with C nodes to obtain an equivalent grammar in Chomsky Normal Form (CNF) (CNF yields parsing algorithms with lower complexity)[2]: $S \rightarrow T\, C \mid T$, $X \rightarrow T\, C \mid T$, $T \rightarrow p$ and $C \rightarrow X \mid X\ X \mid X\ C$. The C nodes can be collapsed and its daughters attached to the parent of C to revert back to the non-binary tree.

While this CFG defines the structure of conversation trees, by itself this grammar is insufficient for our task. In particular, it contains a single non-terminal of each type (S, X, T, C) and so does not distinguish between topics. We extend this grammar to create G_E which has a *set* of non-terminals corresponding to each non-terminal in G_B, these fine-grained non-terminals correspond to different topics. G_E is created using latent annotations (Matsuzaki et al., 2005) on the X, T, S and C non-terminals from G_B. The resulting non-terminals for G_E are $S[i]$, $X[j]$, $T[k]$ and $C[l]$, such that $1 \leq i \leq N_S$, $1 \leq j \leq N_X$, $1 \leq k \leq N_T$, $1 \leq l \leq N_C$. i, j, k and l identify specific topics attached to a particular node type.

Our output trees are created with G_E to depict the topic segmentation of the thread and are non-binary. The binary trees produced by our algorithms are converted by collapsing the C. As a result, conversation trees have $S[i]$, $X[j]$ and $T[k]$

[2] Any context-free grammar can be converted to an equivalent CNF grammar. Our algorithms support unary rules.

nodes but no $C[l]$ nodes.

An example conversation tree for the thread in Table 1 is shown in Figure 1. At level 1, $T[3]$ describes the topic of the first post while the remaining posts are under $X[5]$ which may indicate a *driver* topic, and $X[12]$, a *speaker hardware* topic. Note how $X[5]$ may re-occur in the conversation to accommodate post p_5 on the *driver* topic.

4.2 Supervised learning framework

We use a supervised framework for learning the models. We assume that we have training trees according to the base grammar, G_B. The following section describes our data and how we obtain these G_B-based trees. In Section 6, we present a method for creating G_E-type trees with non-terminal refinements. Estimates of rule probabilities from this augmented training corpus are used to develop the parsers for topic segmentation.

5 Data

We collected 13,352 computer-troubleshooting related threads from `http://forums.cnet.com/`. The number of posts per thread varies greatly between 1 and 394, and the average is around 5 posts. We divide these threads into training, development and test sets. The most frequent 100 words from the training set are used as stopwords. After filtering stopwords, a post contains 39 tokens on average and the vocabulary size of our corpus is 81,707. For development and testing, we only keep threads with a minimum of 3 posts (so that the problem is non-trivial) and a maximum of 50 posts (due to complexity of parsing). We have 9,243 training threads, 2,014 for development, and 2,071 for testing.

A particular feature of the forums on `cnet.com` is the explicit reply structure present in the threads. The forum interface elicits these reply relationships as users develop a thread. When a user replies in a particular thread, she has to choose (only) one of the earlier posts in the thread (including the question post) to attach her reply to. In this way, each post is linked to a unique post earlier in time in the same thread. This reply structure forms a dependency tree. Figure 2 (a) is a possible reply tree for the thread in Table 1.

5.1 Deriving conversation trees

Next we convert these reply-link trees into phrase-structure conversation trees. We developed a deterministic conversion method that uses the generative process defined by the base grammar G_B. The key idea is to track when the conversation branches into sub-topics and when the replies are proceeding within the same topic.

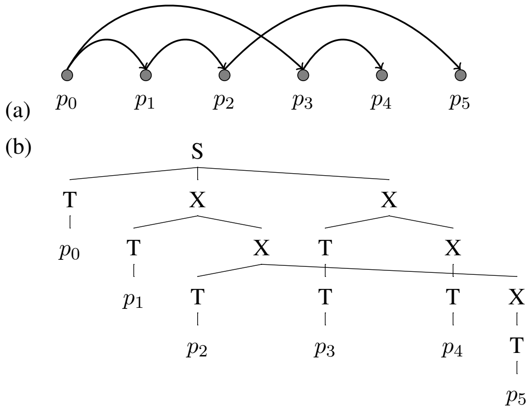


Figure 2: (a) A reply structure tree for the thread in Table 1 and (b) the derived conversation tree

The algorithm traverses the nodes of the dependency tree D in breadth-first order, starting at the root (first) post. We create a root S node in the phrase structure tree H. Then the *thread-starting* rule from G_B, $S \rightarrow T\ X^*$, is used to create one T and k X nodes as children of S. The first post p_0 is attached as a child of the T node. k is equal to the number of replies to p_0 (children of p_0 in D). For each of these k X nodes, we instantiate a $X \rightarrow T\ X^*$ rule in H. The k replies of p_0 are attached one each as a child of the T nodes in these rules. Any set of children are always instantiated in chronological order. So the span of a non-terminal in H always contains posts in non-decreasing time order. We continue the procedure with the next post from D in the traversal order.

This procedure converts the reply tree of Figure 2 (a) into the conversation tree (b). Note that (a) is a possible reply structure for our example thread in Table 1. The conversation tree (b) derived according to this reply structure has a non-projective structure where p_1, p_2 and p_5 are linked under one X node (at level 1). Such a tree can be produced by our LCFRS model. The ideal PCFG tree will repeat the topic branch as in Figure 1.

The derived trees at this stage follow G_B and contain only the S, X, T non-terminals (without any latent annotations). This tree is converted into Chomsky Normal Form using C nodes.

5.2 Discontinuous topic segments

As in our example above, non-projective edges in the *reply structure* are rather frequent. Of the total threads in our corpus 14.5% contain a non-projective edge. A thread should have a minimum of four posts to have the possibility of non-projective edges. Among the 7,691 threads with at least four posts, the percentage of non-projective trees is even higher, 25%. This finding suggests that in any thread of reasonable size which we wish to summarize or categorize, non-projective edges will be common. Hence a direct approach for addressing discontinuous segments such as our PLCFRS model is important for this domain.

6 Parsers for Conversation Trees

The training data are conversation trees with rules from G_B. We refine the non-terminals to create G_E, extract PCFG or PLCFRS rules from the training trees, and build a CYK parser that predicts the most likely tree according to G_E.

6.1 Refining the non-terminals

We use a clustering approach, akin to the spectral algorithm of Cohen et al. (2013) and Narayan and Cohen (2015),[3] to create finer grained categories corresponding to G_B's non-terminals: S, X, C and T. Each node in each tree in the training data is associated with a feature vector, which is a function of the tree and the anchor node. These vectors are clustered (for each of the non-terminals separately) and then, each node is annotated with the corresponding cluster. This process gives us the non-terminals $S[i]$, $X[j]$, $T[k]$ and $C[l]$ of G_E.

The features for a node n_l are: depth of n_l in the tree, root is at depth 0; maximum depth of the subtree under n_l; number of siblings of n_l; number of children of n_l; number of posts in the span of n_l; average length (in terms of tokens) of the posts in the span of n_l; average similarity of the span of n_l with the span of n_l's siblings[4]; similarity of n_l's span with the span of its left-most sibling; elapsed time between the first and last posts in n_l's span.

We use CLUTO toolkit (Karypis, 2002) to perform clustering. The algorithm maximizes the pairwise cosine similarity between the feature vectors of nodes within the same cluster. The best number of clusters for the four non-terminal node types are tuned jointly to give the best performance on our final topic segmentation task.

6.2 Learning rule probabilities

As mentioned previously, each terminal in our grammar is an entire post's text. For the pre-terminal to terminal productions in our grammar $T[j] \rightarrow p_i$, we compute $p(T[j] \rightarrow p_i|T[j])$ as the probability under a unigram language model L_j which is trained on the collection of the posts from the training corpus which are dominated by $T[j]$ nodes. $p(T[j] \rightarrow p_i|T[j]) = \prod_{k=1}^{N_{p_i}} L_j(w_k^i)$ where $w_1^i, w_2^i ... w_{N_{p_i}}^i$ are the tokens in post p_i.

The rest of the production probabilities are learned using MLE on the training trees. In the case of LCFRS rules, the gap information is also obtained during the extraction.

6.3 CYK parsing

For both PCFG and LCFRS we use CYK style algorithms, as outlined in §3, to obtain the most likely tree. For the more computationally complex LCFRS model, we make a number of additions to improve speed. First, we restrict the fan-out of the grammar to 2, i.e. any non-terminal can only span a maximum of two discontinuous segments. 97% of the productions in fact have only non-terminals with fan-out ≤ 2. Second, we use A^* search (Maier et al., 2012) to prioritize our agenda. Last, we reduce the number of items added to the agenda. An item has the form [A, $\vec{\rho}$], A is a non-terminal and $\vec{\rho}$ is the spans covered by A. For every span, we only keep the top 5 non-terminal items according to the score. In addition, we only allow spans with a gap of at most 2 since 77% of all gaps (dominated by fan-out $\leq$ 2) non-terminals are $\leq$ 2 posts. Moreover, after a certain number of items (10,000) are added to the chart, we only allow the creation of new items which have a contiguous span.

7 Systems for comparison

We compare our models to two types of systems.

7.1 STRUCTURE ONLY

The first type generate tree structures without considering the content of the threads.

Right-branching tree (RBT). produces a strictly right branching tree where each post is dominated by the immediately previous (according to time) post in the thread. It uses the grammar

[3] The main difference between our algorithm and the algorithm by Narayan and Cohen (2015) is that we do not decompose the trees into "inside" trees and "outside" trees, or use a singular value decomposition step before clustering the features.

[4] The span of n_l and that of a sibling are each represented by binary vectors indicating the presence and absence of a term in the span. The similarity value is computing using cosine overlap between the vectors and the average across all siblings is recorded.

with the rules $\{S \rightarrow TX$, $X \rightarrow TX$, $X \rightarrow T$, $T \rightarrow p\}$. This method does not perform useful topic segmentation as it produces only a single topic branch containing all the posts.

Attach-to-root tree (ART). attaches each post to the root of the tree. The grammar rules are $\{S \rightarrow TX_1...X_n$, $X \rightarrow T$, $T \rightarrow p\}$, where n is the number of posts in the thread. This approach assumes each post belongs to a different topic in the thread. In contrast to RBT, ART contains too many topic branches, one per post in the thread.

Random tree (RAND). mixes decisions to create a new topic branch or continue in the same branch. The generation process is top down, at each step, the algorithm chooses a certain number of topic branches (Xs) to create ($\leq$ number of posts left to add to the tree). Then, the number of posts under each branch is sampled (such that each branch has at least one post). This process is then recursively done at the new topic branches.

7.2 STRUCTURE AND CONTENT

These approaches produce tree structures informed by content. We build these parsers by modifying prior models for chat disentanglement and linear topic segmentation of documents.

Similarity tree (SIM). produces trees by attaching each post as a child of the most similar of the previous (by time) posts (Wang et al., 2008). We use cosine similarity between vector representations of two posts in order to compute similarity. When the similarity exceeds a threshold value, the post is added under the topic branch of the prior post. Otherwise the post is under a new topic branch attached to the root of the tree. A threshold of 0.15 was chosen after tuning on the development data.

Cluster tree (CLUS). uses an approach related to chat disentanglement (Elsner and Charniak, 2010). The posts within *each* thread are clustered *separately* into k_l clusters where $k_l = l/h$ depends on the number of posts in the thread, l. $h = 6$ was chosen by tuning. The posts in each cluster are ordered by time and a right branching tree is created over them. These k_l cluster-level trees are then attached as children of a new node to create a thread-level tree. The cluster-trees are ordered left to right in the thread-tree according to the time of the earliest post in each cluster.

Linear segmentation tree (LSEG). is based on postprocessing the output of a Bayesian linear topic segmentation model (Eisenstein and Barzilay, 2008). Each post's content is treated as a sentence and a document is created for *each* thread by appending its posts in their time order. The model is then used to group consecutive sentences into k_l segments. For each thread of length l, $k_l = l/h$, $h = 6$ was chosen by tuning. For each segment, a right branching tree is created and these segment-level trees are made siblings in a thread-level tree. The segments are added left to right in the thread-tree as per their order in the text.

All STRUCTURE trees contain thread structure but no topic labels. In other words, they have coarse non-terminals (X, T and S) only. The STRUCTURE AND CONTENT trees, LSEG and CLUS contain topics or groups but only at one top level, and further the number and labels of these topics are different per thread. Hence there is no linking across threads. Within a thread, the SIM and CLUS tree can link non-adjacent posts under the same topic. These links are also not available from a LSEG tree.

8 Evaluation metrics

To evaluate the topic segmentation, we develop a node-governance based measure. Our score compares two conversation trees g and h, where g is the gold-standard tree and h is the hypothesized one. We assume that g and h are in dependency format, reversing the transformation from §5.1.

We break g (and h) into a set of pairs, for each pair of nodes in the tree (each node is a post in the thread). For each such pair, p and q, we find their least common ancestor, $\ell(p, q|g)$ (or $\ell(p, q|h)$. If these nodes are in a governing relation (p dominates q or vice versa), then $\ell(p, q)$ is the dominating node. We then define the following sets and quantities for $\cdot \in \{g, h\}$:

- $S_1(\cdot) = \{(p, q, \ell(p, q)) \mid \ell(p, q|\cdot) \in \{p, q\}\}$.
- $S_2(\cdot) = \{(p, q, \ell(p, q)) \mid \ell(p, q|\cdot) \notin \{p, q\}\}$.
- $n_1(g, h) = |S_1(g) \cap S_1(h)|$.
- $n_2(g, h) = |S_2(g) \cap S_2(h)|$.

$s_1(\cdot)$ and $s_2(\cdot)$ are defined as the size of $S_1(\cdot)$ and $S_2(\cdot)$, repsectively. Let $g_1, \ldots, g_n$ and $h_1, \ldots, h_n$ be a corpus of gold-standard conversation trees and their corresponding hypothesized conversation trees. Then the evaluation metric we compute is the harmonic mean (Fscore) of the micro-average of the precision for governing (G-p) and non-governing (NG-p) pairs, and recall for governing (G-r) and non-governing (NG-r) pairs. For example, G-p is calculated as

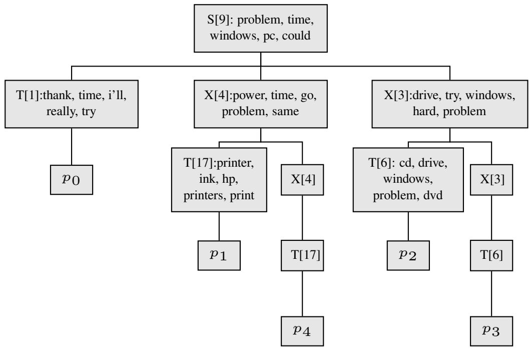
$$\text{G-p} = \frac{\sum_{i=1}^{n} n_1(g_i, h_i)}{\sum_{i=1}^{n} s_1(h_i)}.$$

Traditional parsing evaluation measures such as constituency bracketting and dependency attachment scores were too local for our purpose. For example, if a long chain of posts is placed in a different topic but their local dependencies are maintained, we only penalize one constituent and one node's parent in the constituency and dependency scores respectively. But the topic segmentation created by this change has several posts placed in the wrong topic branch. Our scores overcome this problem by considering the relationship between all pairs of posts and also dividing the relationship in the pair as governing or non-governing.

9 Results and discussion

We tune the number of latent topic annotations for the non-terminals using grid search on the development set. The best settings are 40 S, 100 X, 20 C, 80 T clusters for PCFG and 10 S, 5 X, 15 C, 40 T for LCFRS.

Below we show an example non-projective tree created by our LCFRS parser. The topics are indicated with the most frequent 5 words in the matching cluster.


S[9]: problem, time, windows, pc, could
T[1]:thank, time, i'll, really, try
X[4]:power, time, go, problem, same
X[3]:drive, try, windows, hard, problem
p_0
T[17]:printer, ink, hp, printers, print
X[4]
T[6]: cd, drive, windows, problem, dvd
X[3]
p_1
T[17]
p_2
T[6]
p_4
p_3

Here post p_4 though later in posting time is predicted to be on the same topic as p_1.

The non-terminals in our trees enable useful topic segmentation and we found that performance is extremely sensitive to the number of non-terminals of each type S, X, C and T. Currently, we do not have a direct method to evaluate the non-terminals in our tree but we plan to use the information in other applications as an evaluation.

Table 2 and 3 shows the segmentation performance of the models (as percentages). The performance varied greatly depending on the length of the threads and hence we show the results separately for threads with up to 15 posts (SHORT) and those with 16 to 50 posts (LONG). The results are divided into sections based on the subset of test data on which the evaluation is performed. The first section (R1.) is performance on all threads, (R2.) only on the projective threads in the test data, and (R3.) only on the non-projective threads.

Among the baselines, the Right-branching trees (RBT) or Attaching to the root (ART) have some advantages: the RBT receives 100% recall of the governing pairs and the ART tree has high recall of the non-governing pairs. However, their Fscores are 0. Recall that the RBT contains a single topic branch and hence no useful segmentation is done; ART is the other extreme where every post is put in a separate topic branch. RAND is the average performance of 3 randomly generated trees for each thread. This method has a better balance between branching and depth leading to 33.4 Fscore for SHORT and 21.5 for LONG threads.

The PCFG and the LCFRS models clearly outperform these baselines. The Fscore improves up to 15% over RAND on SHORT and LONG threads. The grammar models also consistently outperform SIM systems.

With regard to CLUS and LSEG, there is a difference in performance between SHORT and LONG threads and based on whether the desired structure was projective or non-projective. On SHORT threads, the grammar models outperform LSEG and CLUS particularly on the projective threads (the LCFRS model has a 22% higher Fscore). On the longer threads however, the CLUS and LSEG models perform best overall and for non-projective threads. CLUS and LSEG directly model the content similarity of posts while the grammar models make many decisions at level of topic nodes. Remember that the clustering is done per thread in CLUS and LSEG compared to using a common set of topics across all threads. Making such fine-grained similarity comparison appears to be helpful especially for longer threads and even though LSEG does not make non-projective decisions, its accuracy is high on the attachments it makes leading to good performance on non-projective threads too. In future work, we plan to explore how we can combine the advantages of direct similarity with the grammar models.

Between the two grammar models, the LCFRS model is better than PCFG, even on projective threads, and can produce non-projective trees. Part of this improvement on projective trees could be due to more data being available in the LCFRS model since all the data can be used for training it.

Model	Ex	G-p	G-r	NG-p	NG-r	F
	R1. On all gold threads (1,971 threads, 24,620 post pairs)					
RBT	20.4	50.8	100.0	100.0	0.0	0.0
ART	5.6	100.0	0.0	42.3	86.0	0.0
RAND	5.2	55.5	19.4	39.2	65.5	33.4
SIM	5.7	68.2	13.3	43.1	79.0	27.9
CLUS	20.2	52.9	85.5	47.5	17.2	42.8
LSEG	20.2	53.0	88.2	52.2	16.5	42.8
PCFG	9.7	52.7	60.4	41.0	34.9	**48.3**
LCFRS	11.4	53.3	62.5	43.6	35.9	**49.9**
	R2. On projective gold threads only					
RBT	24.4	59.8	100.0	100.0	0.0	0.0
ART	6.7	100.0	0.0	35.1	87.4	0.0
RAND	6.2	62.0	22.0	32.4	63.5	35.3
SIM	5.9	73.8	13.9	36.0	79.6	28.4
CLUS	24.2	59.8	90.6	31.3	7.3	26.8
LSEG	24.2	60.1	91.9	35.6	7.6	27.9
PCFG	11.7	61.2	60.5	35.8	36.4	**49.5**
LCFRS	13.5	62.0	64.5	37.6	35.3	**50.8**
	R3. On non-projective gold threads only					
RBT	0.0	39.1	100.0	100.0	0.0	0.0
ART	0.0	100.0	0.0	51.7	84.8	0.0
RAND	0.0	42.0	14.3	47.3	67.3	26.9
SIM	4.3	58.1	12.1	52.1	78.5	26.0
CLUS	0.0	41.2	75.1	54.4	25.8	45.4
LSEG	0.0	41.8	80.7	59.8	24.1	**45.9**
PCFG	0.0	41.1	60.1	47.6	33.5	45.2
LCFRS	0.3	40.8	58.5	50.4	36.4	**46.0**

Table 2: Results on threads with up to 15 posts: for the grammar models (PCFG and LCFRS) and comparison systems (See Section 7). 'Ex' is percentage of fully correct trees and other scores are from Section 8. Top two Fscores are in bold.

Model	Ex	G-p	G-r	NG-p	NG-r	F
	R1. On all gold threads (100 threads, 27,590 post pairs)					
RBT	0.0	21.2	100.0	100.0	0.0	0.0
ART	0.0	100.0	0.0	69.8	88.6	0.0
RAND	0.0	38.9	10.0	65.4	78.4	21.5
SIM	0.0	37.0	8.9	67.2	80.9	19.6
CLUS	0.0	32.6	37.3	73.3	70.5	**42.0**
LSEG	0.0	35.4	50.4	76.8	68.0	**48.4**
PCFG	0.0	24.6	54.1	54.3	36.8	36.6
LCFRS	0.0	22.8	71.4	68.7	29.4	36.5
	R2. On projective gold threads only					
RBT	0.0	36.7	100.0	100.0	0.0	0.0
ART	0.0	100.0	0.0	57.1	90.3	0.0
RAND	0.0	59.9	11.0	56.0	82.6	24.3
SIM	0.0	45.9	8.3	54.4	80.2	19.1
CLUS	0.0	42.0	38.1	60.0	63.2	45.3
LSEG	0.0	51.3	55.0	68.0	65.1	**56.9**
PCFG	0.0	42.2	66.6	34.5	22.9	39.9
LCFRS	0.0	49.0	65.3	51.6	41.7	**52.3**
	R3. On non-projective gold threads only					
RBT	0.0	19.6	100.0	100.0	0.0	0.0
ART	0.0	100.0	0.0	71.1	88.5	0.0
RAND	0.0	36.1	9.9	66.3	78.1	20.9
SIM	0.0	35.8	9.0	68.5	81.0	19.7
CLUS	0.0	31.2	37.1	74.6	71.1	**41.3**
LSEG	0.0	33.2	49.6	77.6	68.2	**46.8**
PCFG	0.0	22.3	51.6	55.8	37.9	34.8
LCFRS	0.0	20.9	72.6	71.6	28.4	34.8

Table 3: Results on threads with $>$ 15 posts

For the PCFG model, only the projective data can be used for training.

Overall, the LCFRS model is powerful on projective threads and SHORT non-projective threads. Compared to PCFG, the LCFRS model has a number of advantages: we can use more data, can predict non-projective trees. Some of the constraints we imposed on the LCFRS parser, such as restricting the gap degree are likely to have limited the ability of the model to generate more flexible non-projective edges. We believe that as we figure out how to make these parsers faster, we will see even more improvements from the LCFRS models.

10 Conclusions

This work represents a first approach to learn discourse structure of forum threads within an explicit grammar framework. We show that a coarse grammar for structure can be refined using latent annotations to indicate the finer topic differences. Our trees have good segmentation performance and provide useful summaries of the thread content at the non-terminal nodes. A main goal for future work is to incorporate further domain specific constraints on the models to improve parsing speed and at the same time allow more flexible trees. We also plan to evaluate the usefulness of conversation trees in tasks such as predicting if a thread is resolved, and user expertise.

Acknowledgements

We thank the anonymous reviewers for their suggestions. We also thank Bonnie Webber, Adam Lopez and other members of the Probabilistic Models of Language reading group at the University of Edinburgh for helpful discussions. The first author was supported by a Newton International Fellowship (NF120479) from the Royal Society and the British Academy.

References

P. H. Adams and C. H. Martell. 2008. Topic detection and extraction in chat. In *Proceedings of the IEEE International Conference on Semantic Computing*, pages 581–588.

E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. 2008. Finding high-quality content in social media. In *Proceedings of WSDM*, pages 183–194.

H. Chen, S. R. K. Branavan, R. Barzilay, and D. R. Karger. 2009. Content modeling using latent permutations. *Journal of Artificial Intelligence Research*, 36(1):129–163.

S. B. Cohen, K. Stratos, M. Collins, D. P. Foster, and L. Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL*.

G. Cong, L. Wang, C. Lin, Y. Song, and Y. Sun. 2008. Finding question-answer pairs from online forums. In *Proceedings of SIGIR*, pages 467–474.

L. Du, J. K. Pate, and M. Johnson. 2015. Topic segmentation with an ordering-based topic model. In *Proceedings of AAAI*, pages 2232–2238.

J. Eisenstein and R. Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of EMNLP*, pages 334–343.

J. Eisenstein. 2009. Hierarchical text segmentation from multi-scale lexical cohesion. In *Proceedings of HLT:NAACL*, pages 353–361.

M. Elsner and E. Charniak. 2010. Disentangling chat. *Computational Linguistics*, 36(3):389–409.

M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing. 2003. Discourse segmentation of multiparty conversation. In *Proceedings of ACL*, pages 562–569.

B. J. Grosz and C. L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, July.

M.A. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of ACL*, pages 9–16.

M. Jeong and I. Titov. 2010. Unsupervised discourse segmentation of documents with inherently parallel structure. In *Proceedings of ACL: Short papers*, pages 151–155.

S. Joty, G. Carenini, and R. T. Ng. 2013. Topic segmentation and labeling in asynchronous conversations. *Journal of Artificial Intelligence Research*, 47(1):521–573.

L. Kallmeyer and W. Maier. 2013. Data-driven parsing using probabilistic linear context-free rewriting systems. *Computational Linguistics*, 39(1):87–119.

G. Karypis. 2002. Cluto - a clustering toolkit. Technical Report TR-02-017, Dept. of Computer Science, University of Minnesota.

S. Kim, L. Cavedon, and T. Baldwin. 2010. Classifying dialogue acts in one-on-one live chats. In *Proceedings of EMNLP*, pages 862–871.

D. E. Knuth. 1977. A generalization of dijkstra's algorithm. *Information Processing Letters*, 6(1):1–5.

R. Levy. 2005. *Probabilistic models of word order and syntactic discontinuity*. Ph.D. thesis, Stanford University.

M. Lui and T. Baldwin. 2009. Classifying user forum participants: Separating the gurus from the hacks, and other tales of the internet. In *Proceedings of the 2010 Australasian Language Technology Workshop*, pages 49–57.

W. Maier, M. Kaeshammer, and L. Kallmeyer. 2012. Plcfrs parsing revisited: Restricting the fan-out to two. In *Proceedings of the 11th International Workshop on Tree Adjoining Grammar and Related Formalisms*.

I. Malioutov and R. Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of COLING-ACL*, pages 25–32.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic cfg with latent annotations. In *Proceedings of ACL*, pages 75–82.

S. Narayan and S. B. Cohen. 2015. Diversity in spectral learning for natural language parsing. In *Proceedings of EMNLP*.

M. Nederhof. 2003. Weighted deductive parsing and knuth's algorithm. *Computational Linguistics*, 29(1):135–143.

A. Nenkova and A. Bagga. 2003. Facilitating email thread access by extractive summary generation. In *Proceedings of RANLP*.

D. Shen, Q. Yang, J. Sun, and Z. Chen. 2006. Thread detection in dynamic text message streams. In *Proceedings of SIGIR*, pages 35–42.

S. M. Shieber, Y. Schabes, and F. C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1&2):3–36.

M. Utiyama and H. Isahara. 2001. A statistical model for domain-independent text segmentation. In *Proceedings of ACL*, pages 499–506.

K. Vijay-Shanker, D. J. Weir, and A. K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of ACL*, pages 104–111.

Y. Wang, M. Joshi, W. Cohen, and C. P. Rosé. 2008. Recovering implicit thread structure in newsgroup style conversations. In *Proceedings of ICWSM*.

L. Wang, M. Lui, S. Kim, J. Nivre, and T. Baldwin. 2011. Predicting thread discourse structure over technical web forums. In *Proceedings of EMNLP*, pages 13–25.