Masters Thesis

Part 3 of the document.

Post Meeting 4 Notes

Tasks:

- Find open subtitles, SRT files (we should have some, check Leverage and other Movies and Shows in our library)
- Run similarity algorithm on sample traces of labeled data
- Consider automated process to expedite our labeling

Next meeting is 19 Oct (Mon) @ 13:00

Labeling Process Notes

We took the SRT files from the Graham Norton Show, which is a sort of late-night host talk show where celebrity guests are invited. It resembles casual conversation with a direction so it's actually very good to use. However, the challenge is that multiple people are always talking, and the subtitles also contain audience reaction, though the guests do interact with audience members as a third collective.
We immediately notice that our labels are a little vague again, but this is to be expected. Here we should note how we're condensing labels, though we should honestly add new ones to be more accurate.
Additionally, in the talkshow, there are places where the closed questions immediately are answered and grouped with the question, so we should be careful to just list those as respond.agree, unless there is a separate question before, then to which we leave it as a question tag… but we should maybe also think about a better solution for this.

Labels that we may need:

- misc - to indicate applause and other things like actions/gestures, instead of using use.social.convention or x.
- give.statement - instead of using give.opinion, since there's a big differential here between a value neutral statement rather than a statement that has an opinion.

- recall - we should have something to indicate that a speech contains quoted speech, which is unique. Or maybe put it under a category of "give.recall"
- relax.atmosphere - we can expand this definition to include statements that are meant as a joke/banter (since the goal is to impact the mood, not necessarily involved with using social conventions) and maybe expand to just exclamations too?

We may also need to consider sub-conversations and how we can make them generic enough to catch multiple speakers in a conversation, i.e: catch trends without specific labeling on who the speaker is. This may be difficult, and perhaps more inaccurate, but something we should think about.

We definitely have to alter the labels and create a more accurate labeling process. Expanding the genre makes it more apparent. Perhaps we need to think about creating a good set of labels that can fit most conversation in different mediums, and from labeling and process determination, we can discover some sub-conversations, and those can help us determine the archetype of conversation (interview show, extended documentary, language exchange, etc) it becomes more apparent.

Post Meeting 5 Notes:

Tasks:

- ~~Finish sub cost~~
- ~~Finish indel cost~~
- Finish clustering
- Have examples confirming the validation of the functions (breaking each step into a separate part might help with this)

Next meeting is Monday, 26 Oct

Post Meeting 6 Notes:

Tasks:

- More formal explanation of the functions
- Move beyond the paper

Next Meeting is Monday, 2 Nov

We've fixed some of the issues with the substitution and indel scores, and they should be for the most part correct (maybe with exception to certain edge cases). Additionally, we've also been able to replicate the table-algorithm for calculating similarity (it's the same algorithm for edit distance, just the opposite). Additionally, we have to account for the fact that traces can have different lengths, so we enforce that the shorter trace be the base comparison.
In the paper, they suggest that insertion is the complement of deletion, but the implementation for our case gets affected when we implement this because of the many cases where there are cases of labels having a severely low insertion score, causing deletion to increase the similarity score too much. For our current purposes, we have listed deletion as a unit 1 similarity cost.

Now, we're moving onwards first to see how we can further add on to the concept of similarity by adding a notion of "sub-conversations." Which are larger trends in the conversation trace than just co-occurrence and context, which seem like intuitively should be tighter and shorter distance similarity measures, whereas a larger presence of similar "archetypes" of conversation should still contribute to similarity. (i.e: for us, our graham norton show traces shouldn't be that similar to the blackpink documentary, but in a sense, they are still interview based traces, and there should be some attention given to that).

So, for our current situation, our similarity metric is displaying that trace1 and trace2 (the two Graham Norton traces) are similar (positive score) whereas trace1 and trace3 (the blackpink documentary) are dissimilar (negative score). However, as we mentioned, there should be some notion where these two are similar, so perhaps we can look to define other metrics to display similarity. (i.e: WHERE are these two traces similar? For us, it would be the interview format, as well as the

idea that speakers display a high amount of recall). We can start building by first defining these trends in the abstract?

Let's maybe start with brainstorming labels that we would choose to define what we intuitively would call a sub-conversation?

> Monologue - maybe something like a long string of give.statements, with some give.opinions inside? (marked by a respond.agree/respond.deny, since a monologue is broken up by the interjection of another person, in this case of speakers not being labeled, we can assume that a respond label is essentially interjection. Things like questions can still be rhetorical)
>
> Making a deeper connection - this would be something along the lines of recall until question or recall until give.statement/opinion or recall until respond.agree/deny for a chain sequence of recalls. The idea is that one person asks a question that is followed by a longer sequence of recalling, which usually indicates sharing a memory, which in polite conversation is how people tend to form a deeper social connection.

Then we can move onto considering how we would discover this longer trend of labels, maybe something like LTL? Or a variation of LTL to check a longer sequence?
Additionally, there is the issue of determining the speaker. Is there a need for us to determine the speaker currently? Because if we separate speakers as labels, then we'll only drive the alignment for similarity of two traces further apart, but if we don't it might be harder to apply notions of longer sequences in an intuitive way? I guess we can say statistically, long sequences of give.statements/give.opinions can be considered a monologue because it's unlikely two speakers interchange give.statements or give.opinions without some sort of interjection. Additionally, the same goes with long sequences of recalls. I guess you could call it human nature. Can we think about something like, recall until question or recall until give.statement/give.opinion/respond.agree/respond.deny. This very linear notion

would fit the bill for something like a question about someone's background, which is a likely candidate to pop up in LE and conversation as a format.

I feel like even looking for sequences for just these two to begin with would already be a difficult task. We can probably just start from there. Finding a long sequence using LTL or some variation of logic is algorithmic, isn't it?

Post Meeting 7 Notes:

Tasks:

- Look at Cost-LTL, which is a LTL variation that works on counters and allows us to have bounds on the number of occurrences in a sequence (useful for things like, "recall should not last more than X utterances).
- Longer term dependencies in sub-conversation discovery that can be used in frequency analysis to further define a similarity metric for our traces!

Our current format (Check Meeting 7 slides) is preferable and good work. We should keep on with this level of explanation and setup for our presentations.

Next meeting is Monday, 9 November

Notes on Cost-LTL related papers

Dimitri mentioned the concept of a LTL variation that is bounded by a cost function. This is something that would be helpful for us in terms of expressing longer-dependency sequences in our system, and have a notion of bounding the number of times a label can occur (via a cost or some value).

From a cursory search, I can't seem to find much beyond the paper, "Linear Temporal Logic for Regular Cost Functions" But it seems relatively recent, and is probably where I should start.

"Linear Temporal Logic for Regular Cost Functions"

- The crux of the argument here is that LTL has good properties, and maintaining those properties while allowing for the extension to counting capabilities is very useful.
- They introduce a new operator $U^{\leq N}$ which implies x $U^{\leq N}$ y means that x holds until y, except at most N times.
- Intuitive and makes sense. We'll proceed and see how they set up this new operator, and how they apply it, and maybe get some insights on how we can approach using this notion in our algorithms to maintain the same notion of counters.
- They suggest their motivation is that if we can bound the number of occurrences of certain outcomes, then we can guarantee the global bounding of the number of "mistakes" (defined events) that occur. This also makes sense when talking about in the context of cost functions.
- For us, our usage of it would seem more on the lines of, we define certain sequences using this bounded metrics for similarity (i.e: long monologues and short monologues are built different). But thinking about it like this, perhaps the bounding in this manner is less important for us. Maybe there is a different bounding that we should be thinking about. Regardless, on face value, this is what we would use it for.
- There are a lot of technical definitions, but the start to the part that we perhaps care more about starts at Definition 4.1, where we outline the extended LTL to describe cost functions. The grammar is listed and contains conjunctions, disjunctions, next (X), until (U), until-N ($U^{\leq N}$). Which tells us that the concepts expressed here can be built using just these notions of operators (which is important for me to know when concerns of implementations are important).
- I get the distinct feeling that this comment will be useful during implementation somewhere, "Remark that we do not need a dual operator for U, because we can use Ω to negate it: $\neg(\phi U \psi) \equiv \neg\psi U(\neg\phi \vee \Omega)$."
- They also have ways to define Eventually (F) and Globally (G) through using untils with TOP or BOTTOM and the Ω alphabet, which stands for the end of the word.

- The goal here for them is to be able to extend LTL such that we can associate a function with the LTL formula.
- Also note that they define that the $U^{\leq N}$ operator always appears positively in the formula, meaning that if (u,n) models φ, then for all k <= n, (u,k) models φ as well. (Which makes sense since if you're bounded by a value, if we increase that value, then you should still be bounded)
- They describe further on the proofs for showing the complexity of the extended LTL formula, but we're not too concerned with that. What we've noted above is sufficient for applying the extension to define subsequences that are of interest to us.

Taking a step back, thinking big-picture

We've done the preliminary implementation of until and until-N, but of course we'd have to modify it for a similarity measurement purpose. However, before that, we would like to take a moment to think and consider the bigger picture, about how what we're currently doing fits into our larger goal.

Of course, our goal is to develop something that would be useful for conversation modeling. We're doing conversation modeling from the perspective of process analysis rather than the linguistic aspects (like NLP). If we can think about it as two sides of the same coin, the coin (goal) of both fields is to create tools that aid in language learning/processing. The NLP side aims to address notions of language from a linguistic sense, and often take ML models to turn words into numbers and values, and attempt to optimize and solve the problem through that method. Our approach is from the field of process mining and model verifications. We see the problem and aim to optimize and solve it through turning words into collection of processes to be analyzed.

Going from that top-level goal, we move down to what we're currently doing. We're determining similarity measures in order to compare traces of conversation to see if we can differentiate them using some metric. We do this because this allows us to attribute some properties onto these traces, such as similarity in purpose (in the conversation) or effect (on the conversation), as well as properties such as containing sub-conversations (that we've defined) which can also be defining factors to these processes. This is all for the express purpose of suggesting that

by applying these notions onto traces of conversations, that we can derive some useful information. Most process analysis work has been largely applied to business mechanics, or in our CS field to robotics (extremely loose interps) and gene-sequencing (extremely strict interps). We believe language conversations are somewhere in the middle (in terms of strictness/looseness). This allows us to tack on top some extra properties/analysis (i.e: sub-conversations, maybe look at probabilities of occurrence of sub-conversations based on trends of sequences, etc). All of this could be then applied to be useful in a larger picture of language processing. (i.e: think about using these models of analysis to detect developed speech vs simple speech of children, has educational purposes).
We're very happy with this direction so far, so we'll keep pushing sub-conversation discovery (which is for what we've seen, relatively new) and all that it could entail (probabilities of occurrence, similarity/differences in resulting models, etc). EXCITING!!!

Preliminary sub-conversation similarity exploration
Wrote the function for Until-N, hopefully to be able to describe sub-conversations of the form: Give.Monologue, and Recall.From.Memory
We get some potential sequences back, we'd have to determine what the N cutoff should be, as well as how accurate we think the returned sequences really match the concept of the sub-conversation we're thinking about.
Additionally, we should have a function to look at relative position of defined sub-conversations to determine their weight when we do comparisons of similarity via sub-conversation frequency contribution.
There's also a consideration to omit/remove short sequences that are only length 2 or 3, since they hardly seem like they would qualify as the long-term dependencies that we're really looking for.

Some properties that could potentially imply good representation?

- Low N violation value, High length?
-

However, seems like from our sample, there isn't many that fit this description. (Meaning there aren't a lot of long runs of "give.statement") but there are some that may fit the bill for recall (which we expect, since we had this idea in mind when looking at the Blackpink documentary trace).

There's the distinct possibility that we can create sets that fit the best properties/representation, and then those results are the "discovered" sub-conversations that we get. (i.e: from the given trace, try different combinations of labels for the sets of expressions for until-N that best fit the properties we want from above, such as low N violation value, with high length)

Starting with a simple strict one-label -> one-label definition, we generate possible sub-sequences for consideration (though we need a function to filter out qualities that we want to look for, such as minimum length and number of N violations).
We can probably expect the strict one-labels to generally not yield results in most cases. (Intuition suggests that this expression would match cases that simply yielded lengths 1, which are essentially X (Next) function matches).
As a general rule, we'll probably only want to expand the combinations for the second set of labels, as that is the one putting the bound on the number of N violations, as well as the length.