NLP Project
Linguistic Repair: Repetition Identification

Team Sheboi

Shelly Jain (20171008)
Sravani Boinepelli (20171050)

Project mentors:

Faculty Coordinator(s): Rajeev Sangal, Dipti Misra Sharma
TA mentor(s): Vandan Mujadia

Introduction

Problem Space

Speech to Speech Machine Translation (SSMT) faces several challenges. There exist many key differences between written and spoken language; during SSMT these differences must be handled before the translation can be done, in order to avoid inaccuracies. Since oral communication (speech) is transient, many sentence constructions are not strictly grammatical. These errors in speech are referred to as disfluencies. Dealing with the disfluencies (in speech transcripts) is referred to as the process of linguistic repair.

Project Description

Disfluencies of a few types can be observed in speech transcripts - hesitations, repetitions, revisions and restarts. Based on the type of disfluency studied, the task of linguistic repair has been broken down into three separate problems: filler word identification, repetition identification and ellipsis identification. Each task of linguistic repair is one of the preliminary tasks required to solve the problem of SSMT. This project tackles repetition disfluencies, limited to only identification of the disfluency and not generation of the repaired sentence. The task is to deduce a single expression to replace the sequence of words where the repetition occurs.

An example of repetition:

> "*I said propulsion now have introduced a new term chemical propulsion chemical aerospace chemical propulsion is what we are going to discuss in this so in this course chemical propulsion*"

Project Objective

The objective of this project is to devise a set of rules to identify instances of repetition disfluencies in speech transcription, as a precursor to the larger task of SSMT.

Tasks

The tasks have been broken down as follows:

1. Task 1 - Identify and analyse the notion of linguistic repair. Devise a few general rules for a system which may be used to identify cases where repetitions occur.
2. Task 2 - Apply the knowledge about occurrences of repetition to create a system to execute the procedure on the provided data. Use the system to prepare tagged data by automatically annotating the dataset with the identified instances of repetition.
3. Task 3 - Assess the quality of the automatic annotation of repetitions in the transcripts. Reformulate the rules for the identification system. Re-evaluate until acceptable level of accuracy is achieved.

Format

Generally, thought process (T), hesitation (H) and stammering (S) cause a speaker to
repeat some of the already spoken segments. With respect to this, the annotation format for the instances of repetition (marked under H) is as follows:

1. Mark entire repeated and final segment with *<REP>* for start and *</REP>* for end.
2. Mark the most informative segment with *<REP-H>* for start and *</REP-H>* for end.

Using the same example of repetition:

> "*I said propulsion now have introduced a new term <REP>chemical propulsion chemical <REP-H>aerospace chemical propulsion</REP-H></REP> is what we are going to discuss in this so in this course chemical propulsion*"

Dataset

The dataset used is the speech transcripts from NPTEL (National Programme on Technology Enhanced Learning) lecture videos of 20-30 minutes each, that have been transcribed by human annotators. The video series selected was the course titled *'NOC: Natural Language Processing'* (Course ID: 106105158). For each video lecture, files in the following format were provided: *<no>_text.txt*, *<no>_nltk.txt* and *<no>.srt*. The task of repetition identification was performed on the *<no>_nltk.txt* files.

Relevant Literature

Several papers were referenced during the course of the project. Certain concepts, features and implementation ideas were adopted from each. The following is a list of the papers which provided the main contribution, along with a brief description of each.

1. Detecting and Correcting Speech Repairs
   This paper presents an algorithm that detects and corrects speech repairs based on finding the repair pattern. The repair pattern is built by finding word matches and word replacements, and identifying fragments and editing terms. The authors do not use a set of pre-built templates, but build the pattern on the fly.
2. A Speech-First Model for Repair Detection and Correction
   The paper identifies several cues based on acoustic and prosodic analysis of repairs in a corpus of spontaneous speech, and propose methods for exploiting these cues to detect and correct repairs. The authors test the acoustic-prosodic cues with other lexical cues to repair identification on a prosodically labeled corpus.
3. Repair Strategies in English Conversations and their Application in Teaching English Interaction Skill to B2 Level Learners
   Based on an investigation into 100 conversations taken from English films which centre on everyday familiar topics, the paper is aimed at presenting some important repair strategies and making some suggestions for applying these strategies to the teaching of the English spoken interaction skill to B2 level learners.
4. Automatic Disfluency Identification in Conversational Speech using Multiple Knowledge Sources
   The paper investigates several knowledge sources for disfluency detection, with different components designed for different purposes. Detection of disfluency interruption points is best achieved by a combination of prosodic cues, word-based cues, and POS-based cues. The onset of disfluency is best found using knowledge-based rules. Specific disfluency types can be aided by modeling word patterns.
5. Correction of Disfluencies in Spontaneous Speech using a Noisy-Channel Approach (Chapter 2: Disfluencies)
   Some statistical MT ideas are adopted for disfluency correction. The disfluent speech is the source language, and fluent speech is the target language. The models are trained on texts manually annotated with disfluencies. Information about the structure and contexts of the disfluencies is extracted. Advantages are that *(1)* extensive linguistic knowledge not required, *(2)* rapidly adapts to new target languages and *(3)* models for unrecognised disfluencies are easily incorporated. Tighter coupling with ASR engines allows the ASR word lattices to be used as input instead of manually transcribed speech.

Methodology

The basic idea behind the procedure to find instances of repetition is to observe a small span of words (dependent on the window of dependency being observed). Within that span of words, repetition is detected and classified. Finally, all the overlapping tags spanning the repeating word segments are disambiguated. This method is modified to account for speaker-specific tendencies.

Ideas and Experiments

All the following examples are real examples taken from the dataset. Certain replacements have been made for the sake of readability:

1. <REP>*text*</REP> with *(text)*
2. <REP-H>*text*</REP-H> with *<text>*
3. All words with single capital letters

Position of the <REP-H> tag

Given input: *H T I A H I A N*; Desired output: *H I A N*

1. Just before final instance
   After initial assignment: *(H T ((I (A <H>) <<I>) A>>)) N*
   After disambiguation: *(H T I A I <A>) N*
   Final repaired sequence: *A N*
2. Just after penultimate instance
   After initial assignment: *(H <T ((I (<A <<H>) I>) A>>)) N*
   After disambiguation: *(H T I A <H I A>) N*
   Final repaired sequence: *H I A N*

Method selected: just after penultimate instance

Disambiguation of the nested tags

Initial decision was made to keep the widest span of repair detected, i.e. the unior of all repair spans. Hence, the first instance of <REP> (here, '*(*') and last instance of </REP> (here, '*)*') were taken and the rest were ignored.

Given input: *C P C A C P*; Desired output: *A C P*

1. Taking the first <REP-H> and last </REP-H>
   After initial assignment: *(C <P ((C>) <<A C>>)) P*
   After disambiguation: *(C <P C A C>) P*
   Final repaired sequence: *P C A C P*
2. Taking the last <REP-H> and last </REP-H>
   After initial assignment: *(C <P ((C>) <<A C>>)) P*
   After disambiguation: *(C P C <A C>) P*
   Final repaired sequence: *A C P*

Method selected: taking the last <REP-H> (here, '<') and last </REP-H> (here, '<')

Size of the n-gram: 4 vs. 5

Given input: *A A B C B A C B*; Desired output: *A C B*

1. Using a 4-gram
   After initial assignment: *(A <A>)((B (<<C (<B>>)) <A C>) B>)*
   After disambiguation: *(A <A>) (B C B <A C B>)*
   Final repaired sequence: *A A C B*
2. Using a 5-gram
   After initial assignment: *(A (<A>) (B (C <B <A>) C>) B>)*
   After disambiguation: *(A A B C B <A C B>)*
   Final repaired sequence: *A C B*

Method selected: using a 5-gram

Speaker tendencies and fixed expressions

Given input: *U N V V S I V V Q M*; Desired output: *U N S I V V Q M*

1. Without using a dictionary
   After initial assignment: *U N (((V (<<V>>)) <<S I ((V>) <<V>>>))) Q M*
   After disambiguation: *U N (V V S I V <V>) Q M*
   Final repaired sequence: *U N V Q M*
2. Adding the dictionary
   After substitution: *U N W S I W Q M*
   After initial assignment: *U N ((W <<S I W>>)) Q M*
   After disambiguation: *U N (W <S I W>) Q M*
   Final repaired sequence: *U N S I W Q M*
   After re-substitution: *U N S I V V Q M*

Method selected: adding the dictionary

Final Implementation

Preprocessing

The given system has been designed to be speaker specific. This may be done as follows:

1. Build a dictionary of expressions frequently used by the speaker which might be wrongly identified by the automated annotation as instances of repetition. For the given dataset, two common examples of this were: '*very very*' and '*really really*'. This will be dependent entirely on the language patterns of the speaker.
2. In the dictionary, also include the general frozen expressions which may be accidentally tagged, like '*over and over*', '*one to one*', '*day to day*', etc.
3. Keep all the dictionary keys for each fixed expression as some character or string which is not found in either the dataset or the tagset. This may be some random sequence of capital numbers or numbers.
4. Replace all instances of dictionary entries (fixed expressions) in the dataset with their corresponding keys in the dictionary.

Run the algorithm for assigning the tags (both initial assignments as well as disambiguation) on this preprocessed dataset.

Algorithm

For initial assignment of the tags:

1. Take an n-gram of some moderate size 'n' (4 or 5). In practice, 5 proved to be the most effective n-gram size.
2. Start checking for repetitions from the first word. If a repeated word is found, mark *<REP>* before the first instance and mark *</REP>* after the last instance. Then mark *<REP-H>* after the penultimate instance, and mark *</REP-H>* between *</REP>* and the end of the last instance.

3. Check for repetition of the next (adjacent) word. If it is repeated, then remove the original `<REP-H>` from after the previous word and move it to after this adjacent word. If the repeated instance is after `</REP>`, then remove both the original `</REP-H>` and `</REP>` and instead mark them after this repeated instance of the adjacent word.
4. Iteratively execute the previous step `(3)` until no adjacent word is repeated in the n-gram.

For disambiguation of the tags:

1. Read through the tagged dataset as a whole. Do not consider each n-gram separately.
2. If the tags are not nested, then do not alter them.
3. If the tags are nested, then do the following:
    - 3.1. Keep the first `<REP>` tag and the last `</REP-H>` tag. Discard all other `<REP>` and `</REP>` tags in between.
    - 3.2. Keep the last `<REP-H>` tag and the last `</REP-H>` tag. Discard all other `<REP-H>` and `</REP-H>` tags in between.

Once the assignment and verification of the tags has been, replace the proxy variables in this tagged dataset with the original fixed expressions. This is the finalised tagged dataset.

Results

The project produced several meaningful results, both expected and unexpected. The main results from the project have been concisely explained in the following sections.

Observations

The repetitions in the dataset, on classifying based on motivation, could be grouped in one of the following nine categories:

1. Expressions of stress - ‘*very very*’, ‘*really really*’, ‘*many many*’
2. Frozen expressions - ‘*day to day*’, ‘*time to time*’, ‘*over and over*’, ‘*one to one*’
3. Hesitation or stammering - ‘*hands on so its its*’, ‘*that your ah that that your planning*’
4. Corrections - ‘*they can they will*’, ‘*they will assignments will be*’
5. Jargon and terminology - ‘*w i minus one w i w i plus one*’ (i.e. $w_{i-1}w_iw_{i+1}$)
6. Mathematical expressions - ‘*zero point zero zero two*’, ‘*this plus this ok*’
7. Expressions of contrast - ‘*then may be same they may not be same now*’
8. Filler words - ‘*so yeah so unless*’, ‘*so ah so*’
9. Extra-linguistic expressions - ‘*this function this function*’ (with gestures/physical cues)

The distribution of repetitions into each of the categories varies with respect to both speaker and domain, as does the individual lexicon of each category. For example, the speaker of the given dataset had a tendency to use a large number of double constructions for emphasis (e.g. ‘*very very*’). Also, since the domain of the transcripts was NLP, there were several common terms in which repetition was observed (like ‘*w i minus one w i w i plus one*’).

Conclusions

Certain trends were noticed within the repetitions that were identified in the dataset. The major inferences from the data were as follows:

1. The major fraction of the repetitions in the dataset were due to content words like determiners and prepositions.
2. Filler words showed frequent repetitions, especially in instances of stammering.
3. Filler words usually occurred at clause boundaries.
4. In the case of corrections, instances were usually of double pronoun occurrences (‘*you we*’), double article occurrences (‘*the a*’) or double preposition occurrences (‘*for of*’). In an extension this can be handled by a POS-based LM which checks for repetition of certain POS categories.

These may be used in an extension of the system which incorporates other systems to handle not only immediate repetition but also other disfluencies like corrections due to revisions and restarts.

The dataset, annotated files and code for parsing may be found at the link:
NLP Project: Linguistic Repair.

Error Analysis

The system designed is not robust. It fails to handle a few cases of repetition, and wrongly labels other cases which aren’t truly examples of repetition:

1. Filler words - In some cases, filler words are used as ellipses and the words preceding it are repeated afterwards as well. This is especially common in the case of ‘*so*’, as seen in the given dataset, because it functions as both filler word and function word. This might be improved by incorporation a POS-based LM.
2. Speaker habits - Difficult to determine which repetitive constructions are speaker patterns and which are just repetition, without comparing the transcript to the actual speech. For automated tasks this is not possible. Additionally, the dictionary built to handle such cases must be manually done, which is also not feasible. This could possibly be automated by incorporating a LM that determines which constructions are more likely.
3. Terminology - Like in the case of the speaker habits, these have to be manually added to a dictionary. Again, these may be handled by an LM.
4. Conflicting expressions - In some cases, the terminology my look identical to repetition in an ordinary construction (like, ‘*i i*’ vs. ‘*i minus one i i plus one*’ or ‘*f one to f six*’). In such cases the current system will treat it as either of the two in all cases, depending on the dictionary entries. This can only be handled using extensive exceptional cases or an additional LM.
5. Specifically framed expressions - When looking at certain types of expressions, especially mathematical expressions, deliberate constructions can be mistaken for repetitions. Some examples are ‘*this plus this ok*’ and ‘*zero point zero zero two*’. Other examples of this are comparisons, in which terms are necessarily repeated for a contrastive effect (e.g. ‘*then may be same they may not be same now*’).

6. Length of the n-gram - Currently, the system is heavily dependent on the length chosen for the n-gram. For this system, optimal length was experimentally verified, but there are still possible cases of conflict due insufficient/excessive length. This issue can only be handled by using an artificial neural network that determines the long-term dependencies in the discourse and provides an ideal n-gram length which preserves this dependency without also preserving the repetitions.
7. Extra-linguistic knowledge - Since the dataset consists of speech transcripts, details like gestures facial expressions and other non-linguistic cues are not preserved. It is impossible to determine computationally which sequences are repetitions and which were intentional but differentiated by other cues (e.g. '*this function this function*' with the speaker pointing to a diagram). This limitation can only be overcome if some other form of input is provided along with the transcripts.
8. Quality of the dataset - Accurate annotation of the data is wholly dependent on the correct transcription of the original speech. This must be rectified before the system is applied.

Extensions

The performance of this project may be further improved by incorporating other models and using correlated as well as unrelated features to bring greater discriminative ability. Some possible models which may be incorporated are:

1. POS-based LM - This will be able to detect and predict constructions which require repair by learning the tendencies of occurrences of the different types of disfluencies.
2. Acoustic model - This will identify cues in the speaker's voice signal which indicate the onset of some disfluency.

The combination of the three will be a much more robust model, able to handle a greater variety of disfluencies with less error.

Paper Presentation

Paper which is being analysed is the following:

Automatic Disfluency Identification in Conversational Speech using Multiple Knowledge Sources, Yang Liu, Elizabeth Shriberg and Andreas Stolcke

This work investigates a number of knowledge sources for disfluency detection, including acoustic-prosodic features, a language model (LM) to account for repetition patterns, a part-of-speech (POS) based LM, and rule-based knowledge. Different components are designed for different purposes in the system. Results show that detection of disfluency interruption points is best achieved by a combination of prosodic cues, word-based cues, and POS-based cues. The onset of a disfluency to be removed is best found using knowledge-based rules. Specific disfluency types can be aided by the modeling of word patterns.

Three types of disfluencies are studied in the paper. '*' denotes the right edge of the reparandum region and is called the interruption point (IP).

1. Repetitions - the speaker repeats some part of the utterance.

For example: '*I * I like it.*'

2. Revisions (content replacement) - the speaker modifies some part of the utterance.
   For example: '*We * I like it.*'
3. Restarts (false starts) - a speaker abandons an utterance or constituent and then starts over.
   For example: '*It's also * I like it.*'

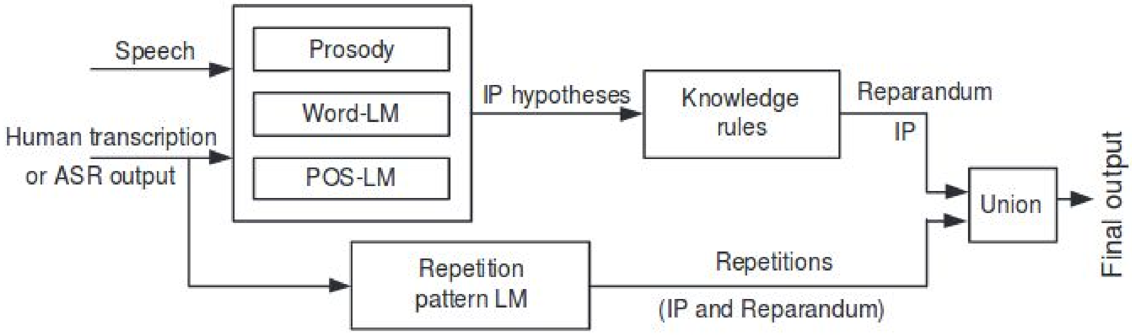

Figure: System diagram

Several of the observations made during the course of the project were in line with the results of the paper. This included the characteristics of the observed repetitions, the most-effective size of the n-gram (here, seen to be 4 or 5) and even the hypothesis of which types of repetitions tend to be most frequent.