UNIVERSITY OF GOTHENBURG

# Project: Non-lexical sounds in dialogue utterances

Student name: *Dominik Künkele*

Course: *Machine Learning 2 (LT2326)*
Due date: *November 30th, 2022*

## Background

In one of their papers, Edlund and his colleagues describe human interaction with machines using metaphors on a spectrum. On the one side of the spectrum is the so called *interface metaphor*. The interface of the machine is built in a way that users are aware, they are talking to a machine. Consequently, they also adapt their language, and use more command like utterances (e.g. "Call John", "Set the timer") as they would fill slots in an imaginary web form. On the other end of the spectrum resides the *human methapor*. Here, the users don't really know, they are talking to a machine and therfore use a "normal" language in the sense of utterances, they would uses in a human-human dialogue. While the interface metaphor is still used very commonly, many dialogue systems like *Alexa* or *Siri* lie somewhere in between these extremes. The human metaphor is implemented rarely and is seen more often in science fiction.

Nevertheless, there are a few strong reasons for using the human metaphor and let the users talk in natural language. Natural language is

**easy to use.** Since we use natural language all the time in all human interactions, it is very natural for us, to also use it in machine interaction.

**flexible.** Natural language allows us to express everything we want to express. We can express for instance thoughts, feelings or facts with different certainities or also for example talk about things that were, things that are, and some things that have not yet come to pass. There are only very few things in a human mind that cannot be represented in natural language (partly in combination with mimic and gestures).

**resilient to error handling.** Furthermore, it allows us, to correct things we said or also specify certain parts more if we realize the listener is not understanding very easily. In the role of the listener, we can also verify, what we understood. Both can be done in various ways, by for instance rephrasing the utterance, repeating all or parts of it or even ask questions about it.

**enjoyable** Using Natural Language is finally also much more enjoyable than using a command like language. This of course depends on the person, but in general, humans evolved to use and interact in this language.

The goal is now, not to create a machine that is very close to humans, but to create "[a] machine that acts human enough that we respond to it as we respond to another human" as Cassel puts it in his paper in 2007.

There are endless factors that could define, *what* human-like interaction is. They inlcude for example a change in pitch (upwards, to signal turn-keeping; downwards, to signla turn-yielding) or choices of words ("This is the automated booking system" will obviously signal the user that it speaks to a machine, while "Hello" does not). One big factor are non-lexical sounds (NLS) like hesitations ("uh", "hmm", ...) or repetitions of words/n-grams ("if I'm home then I I definitely watch her"). This could be famously observed in the *Google Duplex* demo in 2018, when a machine was calling a hair saloon using NLS.

This project aims to make machine utterances more human-like, by adding NLS to an utterance at natural positions. Additionally, the utterances could also be enhances, by adding repetitions of n-grams.

## Data resource

As a resource for this project, I use the *Switchboard corpus*. There are various dialogue corpora published, but the Switchboard corpus contains NLS in a mostly structured way, while keeping the rest of the dialogue mostly undistilled, but annotated and as close as possible to the recordings. This means that the corpus inludes

**non-grammatical sentences:** "yeah i think it is too it's gonna get better"

**repetitions of n-grams:** "oh she didn't she didn't do something"

**annotated partial words:** "bec- because we're so"

**anomalies of words with correct word** "bettle/better"

Furthermore, the corpus classifies the NLS based on there sound into thirteen classes: ah, eh, hm, huh, huh-uh, hum-um, ooh, uh, uh-huh, uh-hum, uh-oh, um, um-hum. These classes will then be used for predicting the NLS in the generated sentences.

I cleansed the corpus, by removing annotations, *silences* and laughters, replacing partial with complete words and NLS with a specific token per class. In the end, I could extract 247 123 utterances. The length of these utterances ranges from 1 to 81 words. Since I only used each utterance by itself to generate NLS and was not using any realtions in the dialoue, I also excluded utterances shorter than than words. That yielded me around 150 000 usable utterances. Around 70 000 of these utterances contain at least one NLS.

Almost half of the utterances contain a repetition. Most of the repetitions are only on word, but I could identify also repetitions of up to 9-grams.

## Methods

The idea of this project was to predict if and which NLS should follow each word in a sentence. The model consists of (A) an embedding layer for tokens and POS, (B) encoder layers for tokens and POS and (B) a classifier that classifies the NLS for the output of the encoding.

In the beginning, the embeddings for tokens and POS were trained from scratch. Later, I switched to pretrained *GloVe* embeddings for the tokens that are based on 6 billion tokens with 300 dimensions. During training, they are frozen until epoch 10, from when they are fine-tuned.

In the project, I wanted to compare different encoding layers and their combination and see, how they affect the predictions. For this, I use (1) a bidirectional LSTM encoder for the tokens, (2) a bidirectional LSTM encoder for the POS and (3) a transformer encoder. The encoders are combined in the following ways:

1. $concat(token\_LSTM, POS\_LSTM, transformer)$
2. $concat(token\_LSTM, POS\_LSTM)$
3. $token\_LSTM + POS\_LSTM$
4. $token\_LSTM + 0.5 * POS\_LSTM$
5. $token\_LSTM$

The classifier finally combines multiple linear layers combines with non-linear functions, to reduce from the output dimension of the encoder to 14 classes (NLS tokens + <NO_NLS> token).

Dropout with a probability of 0.2 is applied to all of the layers during training.

For the sentence *uh-huh i think so and uh*, the representation for the model would look like this:

| input | <SOS> | i | think | so | and | <EOS> |
|---|---|---|---|---|---|---|
| output | uh-huh | <NO-NLS> | <NO-NLS> | <NO-NLS> | uh | <NO-NLS> |

For this, I first tokenized the sentences, using *nltk*s *word_tokenize* function. Afterwards, I extracted the NLS, added start-of-sequence and end-of-sequence tokens and aligned it with the NLS. The input was then encoded in two ways. For the first, I built up my own vocabulary from the train dataset used it for the encoding. These encodings would be used to train word embeddings from scratch. For the second, I encoded the sentence in the vocabulary of *glove*. I used the pretrained glove vectors based on 6 billion tokens using 300 dimensions. I added the special tokens <*SOS*>, <*END*>, <*PAD*> and <*UNK*> with random intializations to its vocabulary. Furthermore, I tagged the parts of speech for each sentence. These were encoded in its own vocabulary. POS embeddings are learned from scratch.

**Results**

Evaluating the models was quite difficult. Since, it is not a standard classification task, but generation of sequences, classic metrics like *accuracy*, *precision* or *recall* may not yield meaningful value. The whole problem could be more seen as a translation task from 'English without NLS' to 'English with NLS'. That's why, I also inlcuded the *Meteor score*, which is usually used in Machine Translation. Still, compared to Machine Translation, the difference between the source and the target sentence is very small, in most of the cases just one token. The Meteor score will therefore be very high and differences between models may not be seen. Furthermore, the placement of NLS (as well as the type of NLS) in a sentence may seem correct for human judgement for multiple positions. This effect is even stronger than in Machine Translation, since NLS carry fewer semantic meaning than actual misplaces words. Consider the following two sentences:

(1) that's contrary to **uh** popular belief you know

(2) that's contrary **uh** to popular belief you know

Sentence 1 is picked from the training data. If the model predicted Sentence 2, all of the metrics would punish the model even though for humans, both sentences sound acceptable. In the end, for this problem, all of the metrics can only give pointers and hints, which model might perform better than others. A human evaluation would be necessary to judge the models performance

I tested the models on two different datasets. In one dataset, I kept all of the utterances, independently if they contained NLS. The test split consists of around 36.8% utterances that contain at least one NLS. Using this data, the models consistently generated very few utterances with NLS (only 4.4% to 5%). For this reason, I created another dataset, that filtered out all utterances without NLS. With this, the best model also generated for around 41% of the test samples an utterances with NLS. The metrics are not directly comparable, because the test sets differ.

For the unfiltered dataset, all of the models produce very high metrics. The accuracy, as well as the Meteor score don't differ significantly across the models. Also the margin for the precision and recall lie under 0.6% and 0.8% respectiveley, which doesn't seem very significant.

| | contain NLS | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | gold | predicted | $\frac{NLS}{<NO\_NLS>}$ | Accuracy | Precision | Recall | Meteor | NLS score |
| (1) | 100 | 41.31 | 6.51 | 90.32 | 47.57 | 50.03 | 93.55 | 48.37 |
| (1) | 36.80 | 4.63 | 2.70 | **96.79** | **79.19** | **80.64** | 97.16 | **49.70** |
| (2) | 36.80 | 4.45 | 2.70 | 96.70 | 78.59 | 79.97 | 97.18 | 49.66 |
| (3) | 36.80 | **4.98** | 2.70 | 96.67 | 78.52 | 79.88 | **97.20** | 49.65 |
| (4) | 36.80 | 4.63 | 2.70 | 96.69 | 78.56 | 79.93 | 97.19 | 49.66 |
| (5) | 36.80 | 4.86 | 2.70 | 96.66 | 78.46 | 79.82 | 97.19 | 49.64 |

Table 1: Evaluation of each model

My model has very high scores for all the metrics *Accuracy*, *Precision* and *Recall*, even though the predicted sentences are not really good. The reason for that is only a very small proportion of the slots are NLS, while the biggest proportion is the <NO_NLS> token. Therefore, my model that often only predicts <NO_NLS> tokens for all of the slots performs very well. To make this problem better visible, I introduced a new metric, called the *NLS score*. It is a weighted accuracy that weights <NO_NLS> tokens antiproportinally to its occurence in the test corpus. More specifically the weights are calculated as following for the whole test corpus:

$$W_{NLS} = \frac{number\ of\ NO\_NLS\ tokens}{number\ of\ slots} \tag{1}$$

$$W_{NO\_NLS} = 1 - W_{NLS} \tag{2}$$

The weights are then applied to each token $t$ of a sentence, depending the gold label. If the gold label was a <NO_NLS> token, $W_{NO\_NLS}$ is applied, otherwise $W_{NLS}$. The sum of the weighted scores is then normalized over the length of the sentence $n$ and averaged over all sentences.

$$A_{W\ sentence} = \frac{W_{NLS} * \sum^{t} correct\ NLS + W_{NO\_NLS} * \sum^{t} correct\ NO\_NLS}{t} \tag{3}$$

$$A_W = \frac{\sum^{n} A_{WS}}{n} \tag{4}$$

Finally, the NLS score is the mean of the resulting weighted accuracy with the average 'normal' accuracy.

$$NLS\ score = \frac{A_W + A}{2} \tag{5}$$

This new metric takes the inbalance of <NO_NLS> and NLS tokens better into account. It is also flexible in respect to a changing ratio, since it could also handle the opposite case if the biggest portion of slots were NLS tokens.

As expected the NLS score is much lower for all different models.

**Discussion**