

# 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 [1]. 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 metaphor*. Here, the users don't really know, they are talking to a machine and therefore use a "normal" language in the sense of utterances, they would use 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 certainties 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 [2].

There are endless factors that could define, *what* human-like interaction is. They include for example a change in pitch (upwards, to signal turn-keeping; downwards, to signal 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 enhanced, by adding repetitions of n-grams.

### Data resource

As a resource for this project, I use the *Switchboard corpus* [3]. There are various dialogue corpora published, but the Switchboard corpus contains NLS in a mostly structured way. The rest of the dialogue is kept mostly undistilled with annotated changes and normalizations. It tries to be as close as possible to the original recordings. This means that the corpus includes

**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 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 relations in the dialogue, I also excluded utterances shorter than two 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 one 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 (C) a classifier that classifies the NLS for the output of the encoding at each encoding step.

**(A) Embedding layer.** In the beginning, the embeddings for tokens and POS were trained from scratch. Here, I used my own built-up vocabulary from the training data, containing also the special tokens for START, END, UNKNOWN and PADDING. With this, I also could vary the embedding size. Later, I switched to pretrained *GloVe* embeddings for the token to help the model to understand the relation between the words easier from the start. I utilized the *GloVe* embeddings that are based on 6 billion tokens embedded in 300 dimensions. I added the special tokens <SOS>, <END>, <PAD> and <UNK> with random initializations to its vocabulary. Since the embeddings are already pretrained, while the rest of my model is randomly initialized, I froze them for the first 10 epochs. After that, they are fine-tuned together with the rest of the model.

**(B) Encoding layer.** The next step was to encode the sentence step by step (word by word). The simplest solution would have been, to just use a LSTM layer and encode the embeddings automatically. In the project however, I wanted to see, how the model performs if I feed it

different kind of information and encode them differently. In especially, I wanted to see if provide the model explicitly syntactical information by giving it the parts of speech (POS).

Therefore, the model is able to receive word embeddings and POS embeddings. Both embeddings are then encoded (1) a bidirectional LSTM encoder and (2) a bidirectional LSTM encoder for the POS. Furthermore, the word embeddings are additionally encoded with (3) a transformer encoder. These layers are then combined for each step in an LSTM by either concatenation or addition, optionally weighted as following:

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

**(C) Classifier.** The classifier is built on top of each step of the LSTMs. This allows me to predict an NLS token in every position of the sentence. For this, the classifier finally combines multiple linear layers together with non-linear functions. This reduced the dimension from the output dimension of the combination of the encoders to 14 classes (NLS tokens + <NO-NLS> token). The number of encoder dimensions differs depending on the way, the encoders are combined. If the number was very high (e.g. for the concatenation of both LSTMs and the transformer), I introduced an additional linear layer.

For all the different layers, I applied dropout with a probability of 0.2 during training. This should prevent the model from overfitting to the training data.

**Input for the model.** As explained before, I needed to feed the model the words aligned with POS. The example sentence *uh-huh i think so and uh* should be represented like this with an output by the model:

input tokens	<SOS>	i	think	so	and	<EOS>
input POS	<SOS>	NN	VBP	RB	CC	<EOS>
output	uh-huh	<NO-NLS>	<NO-NLS>	<NO-NLS>	uh	<NO-NLS>

For this, I first tokenized the sentences, using *nlts word\_tokenize* function. Afterwards, I excluded the NLS, added start-of-sequence and end-of-sequence tokens. This is then aligned with the extracted NLS enriched with <NO-NLS> tokens if there is no NLS at a specific position.

## Results and Discussion

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 included 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 always be very high and differences between models may not be significant looking at this metric.

Furthermore, the placement of NLS (as well as the type of NLS) in a sentence is very subjective. NLS don't carry usually as much semantic meaning as normal words. Additionally, there are no grammar rules, when NLS may be used or not. This leads to the fact that every human

may use these NLS differently and there is now objective ‘truth’ to their placement (which is rather the case with a normal grammar). Misplaced NLS by the model based on the training and test data may therefore be actually not misplaced based on human judgement. 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 the metrics would punish the model even though for humans, both sentences sound acceptable. In the end, all the metrics can only give pointers and hints for this problem, which model might perform better than others. A human evaluation would be definitely necessary to judge the models’ performance better.

	contain NLS		$\frac{NLS}{\langle NO\_NLS \rangle}$	Accuracy	Precision	Recall	Meteor	NLS score
	gold	predicted						
(1)	100	41.31	6.51	90.32	47.57	50.03	93.55	48.37
(1)	36.80	4.63	2.70	<b>96.79</b>	<b>79.19</b>	<b>80.64</b>	97.16	<b>49.70</b>
(2)	36.80	4.45	2.70	96.70	78.59	79.97	97.18	49.66
(3)	36.80	<b>4.98</b>	2.70	96.67	78.52	79.88	<b>97.20</b>	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

I tested the models with two different datasets. In one dataset, I kept all the utterances, independently if they contained NLS. For this, the test split consists of around 36.8% utterances that contain at least one NLS. In the second dataset, I filtered out all sentences that don’t contain any NLS. The proportion is therefore 100%. Since the test data differs from dataset to dataset, the metrics achieved with them can’t be compared directly.

The reason for this was that the models generated almost no NLS at all using the unfiltered dataset. The results consisted only of 4.4% to 5% of sentences that contain NLS (compared to 36.8% of the gold data). Using the filtered data set, 41% of the generations contained NLS (compared to 100% of the gold data).

For the unfiltered dataset, all 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% respectively, which doesn’t seem very significant.

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  $\langle NO\_NLS \rangle$  token. Therefore, my model that often only predicts  $\langle NO\_NLS \rangle$  tokens for all 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  $\langle NO\_NLS \rangle$  tokens anti-proportionally to its occurrence in the test corpus. More specifically the weights are calculated as following for the whole test corpus:

$$W_{NLS} = \frac{\text{number of } \langle NO\_NLS \rangle \text{ tokens}}{\text{number of slots}} \quad (1)$$

$$W_{NO\_NLS} = 1 - W_{NLS} \quad (2)$$

The weights are then applied to each token  $t$  of a sentence, depending on the gold label. If the gold label was a  $\langle NO\_NLS \rangle$  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.

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

$$A_W = \frac{\sum^n A_{WS}}{n} \quad (4)$$

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

$$NLS\ score = \frac{A_W + A}{2} \quad (5)$$

This new metric takes the imbalance 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. Furthermore, it lets me compare the filtered and unfiltered dataset a little better.

As expected the NLS score is much lower for all different models. But also here, the differences between the models are not significant. All models consistently produce too few sentences with NLS. Even using the filtered dataset yields a similar low NLS score.

Next, I looked at the predicted sentences that contained NLS. Most of the NLS, that were predicted were 'uh' or 'um'. It was hard to see any pattern, in which position they appeared. They appeared mostly in the middle of the sentence or towards the end. However, many of them were predicted before a noun or after conjunctions like 'that' or 'but'. Still, there appeared also a lot of NLS, which I couldn't classify together and understand, why they may have been predicted.

## Conclusion

Currently, the model doesn't produce sentences with NLS consistently. There are multiple ways, how the model could be improved in the future. One option could be to change the calculation of the loss. Since the cross entropy, I currently use is affected by the above described problem of imbalance of NLS and <NO\_NLS> tokens, a different calculation may be a better basis for the loss.

Furthermore, the dataset could be improved. First, I could use a bigger dataset, that represents a wider range of utterances and uses of NLS. Secondly, the data could be cleansed from unnecessary parts. Right now, the data holds mostly ungrammatical sentences with repetitions of n-grams, unsyntactical constructions and misspelled words. While this represents, how humans talk, it is maybe not the best basis to translate from grammatical correct 'utterances without NLS' to 'utterances with NLS'. Therefore, it could be attempted, to remove the ungrammaticalities and just leave the NLS. This could provide better results for multiple reasons:

- less unknown words and therefore better embeddings
- patterns in one utterance may repeat in more other utterances, since they are normalized with grammar rules. The model may detect these patterns easier
- POS tagger works better, since they are trained on grammatical sentences

In a next step, the model can be adapted. Right now, I am using a straight forward model, concatenating different encodings and running them through linear layers. Another option could be for example to use a transformer and masking the NLS tokens. Going on step further, I could also try to take the surrounding context into account, i.e. the previous utterance of

the dialogue partner. NLS in a sentence especially in the beginning may depend on what the dialogue partner said and what was understood. This could yield better results, since the model would have more information to analyze and detect patterns.

In the end, it may also be helpful to make a deeper analysis of NLS and see, where they actually appear and why. The results of this analysis could help to build a more targeted model.

For the future, it may also be worth to have a look into silences and repetitions of phrases. These properties seem to be correlated with NLS and depend on each other. In the dataset, an NLS for example often occurs between repetitions. Using this knowledge may give better results for predicting human-like language.

### References

- [1] Edlund, J., Gustafson, J., Heldner, M., Hjalmarsson, A. (2007) *Towards human-like spoken dialogue systems*.
- [2] Cassell, J. (2007) *Body language: lessons from the near-human*.
- [3] ISIP (2003) *Switchboard Corpus*, <https://isip.piconepress.com/projects/switchboard/>.