Karlsruhe Institute of Technology

Backchannel Prediction for Conversational Speech Using Recurrent Neural Networks

Bachelor's Thesis of

Robin

At the Department of Informatics
Institute for Anthropomatics and Robotics
Interactive Systems Labs

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	?
Advisor:	Markus Müller

Duration: ??. Monat 20?? – ??. Monat 20??

Contents

1. Introduction

Motivation, Goals

2. Related Work

Collection:

- Watanabe and Yuuki (1989)
  - not found
- Okato et al. (1996)
  - Languages: Japanese
  - Truth type: utterances
  - Features: Pause
  - Method: HMM for pitch contour
  -
  - Evaluation Method:
  - Margin of Error: (-100ms, +300ms) from target utterance end (?)
- Ward (1996)
  - Japanese
- Noguchi and Den (1998)
  - Languages: Japanese
  - Features: Prosodic (pause, frequency F0)
- Ward and Tsukahara (2000)
  - English, Japanese
  - Features: Low Pitch Cue, Pause
  - Margin of error: (-500, 500)
- Cathcart, Carletta, and Klein (2003)
  - English
  - Features: trigrams, pauses

  - Corpus: HCRC Map Task Corpus
  - Eval Method: Precision, recall, F1

- Fujie, Fukushima, and Kobayashi (2004)
  - Japanese
  - Features: utterances, prosodic

- M. Takeuchi, Kitaoka, and Nakagawa (2004)
  - Japanese
  - features: porosodic
  - Method: decision tree, C4.5 learning algorithm
  - Corpus: SIG of Corpus-Based Research for Discourse and Dialogue, JSAI, 1999. ”Constructing a spoken dialogue dorpus as sharable research resource”
  - Eval method: recall, precision

- N. Kitaoka et al. (2005)
  - Japanese
  - Pitch, pause,
  - Eval: precision, recall, F1

- Nishimura, Kitaoka, and Nakagawa (2007)
  - Japanese
  - Features: Speech recog,

- L.-P. Morency, Kok, and Gratch (2008)
  - English
  - Features: Eye gaze, low pitch, pause
  - HMM, CRF
  - Margin of error: happens during actual BC utterance

- De Kok and Heylen (2009)
  - English
  - Features: dialog, attention, head gestures, prosody (pitch, pause, etc)
  - Margin: peak in our probabilities (see Section 3) occurs during an actual end-of-speaker-turn.

- L.-P. Morency, de Kok, and Gratch (2010)
  - Dutch
  - Corpus: MultiLis corpus
  - Special: building consensus Fconsensus

- de Kok et al. (2010)
- Huang, Morency, and Gratch (2010)
  - Subjective, on live corpus
  - Fconsensus

- Ozkan and Morency (2010)

  - RAPPORT dataset

- Ozkan, Sagae, and Morency (2010)
- R. Poppe et al. (2010)
- de Kok, Heylen, and Morency (2013)
- D. Ozkan and Morency (2013)
- de Kok, Poppe, and Heylen (2014)
- Mueller et al. (2015) # Backchannel Prediction {#sec:extraction}

A listener backchannel is generally defined as any kind of feedback a listener gives a speaker as an acknowlegment in a primarily one-way conversation. They include but are not limited to nodding (Watanabe and Yuuki 1989), a shift in the gaze direction and short phrases. Backchannels are said to help build rapport , which is the feeling of comfortableness or being "in sync" with conversation partners (Huang, Morency, and Gratch 2011).

(-> motivation) This thesis concentrates on short phrasal backchannels consisting of a maximum of three words. We try to predict these for a given speaker audio channel in a causal way, using only past information. This allows the predictor to be used in an online environment, for example to make a conversation with an artificial assistant more natural.

2.1 BC Utterance selection

The definition of backchannels varies in literature. There are many different kinds of phrasal backchannels, they can be non-commital ("uh huh", "yeah"), positive/confirming ("oh how neat", "great"), negative/surprised ("you're kidding", "oh my god"), questioning ("oh are you", "is that right"), et cetera. To simplify the problem, we initially only try to predict the trigger times for any type of backchannel, ignoring different kinds of positive or negative responses. Later we also try to distinguish between a limited set of categories.

2.2 Training area selection

We generally assume to have two seperate audio tracks, one for the speaker and one for the listener each with the corresponding transcriptions. We need to choose which areas of audio we use to train the network. We want to predict the backchannel without future information (causally / online), so we need to train the network to detect segments of audio from the speaker track that probably cause a backchannel in the listener track. The easiest method is to choose the beginning of the utterance in the transcript of the listener channel as an anchor t, and then use a fixed context range of width w before that as the audio range to train the network to predict a backchannel $[t - w, t]$. The width can range from a few hundred milliseconds to multiple seconds. We feed all the extracted features for this time range into the network at once, from which it will predict if a backchannel at time t is appropriate. This approach is easy because it only requires searching for all backchannel utterance timestamps and then extracting the calculated range of audio. It may not be optimal though, because the delay between the last utterance of the speaker and the backchannel can vary significantly in the training data. This means it is not guaranteed that the training range will contain the actual trigger for the backchannel, which is assumed to be the last few words said by the speaker, and even if it does the last word will not be aligned within the context. This causes the need for the network to first learn to align it's input, making training harder and slower.

Another interesting anchor is the last few words before a backchannel. We could for example choose t as the center of the last speaker utterance before the backchannel, and then use $[t-0.5s, t+0.5s]$ as the training range. This proved to be hard because without manual alignment it isn't clear what the last relevant utterance even is, and in many cases the relevant utterance ends after the backchannel happens, so we would need to be careful not to use any future data. The first approach seemed to work reasonably well, so we did not do any further experiments with the second approach.

We also need to choose areas to predict zero i.e. "no backchannel" (NBC). This should be in about the same amount as backchannel samples, so the network is not biased towards one or the other. To create this balanced data set, we can choose the range a few seconds before each backchannel as a negative sample. This gives us an exactly balanced data set, and the negative samples are intuitively meaningful, because in that area the listener explicitly decided not to give a backchannel response yet, so it is sensible to assume whatever the speaker is saying is not a trigger for backchannels.

2.3 Feature selection

The most commonly used audio features in related research are fast and slow voice pitch slopes and pauses of varying lengths. (Ward and Tsukahara 2000; Truong, Poppe, and Heylen 2010; L.-P. Morency, de Kok, and Gratch 2010). Because our network does automatic feature detection, we simply feed it the absolute pitch and power (volume) values for a given time context, from which it is able to calculate the pitch slopes and pause triggers on its own by substracting the neighboring values in the time context for each feature.

We also try to use other tonal features used for speech recognition in addition and instead of pitch and power. The first feature is the fundamental frequency variation spectrum (FFV) (Laskowski, Heldner, and Edlund 2008), which is a representation of changes in the fundamental frequency over time, giving a more accurate view of the pitch progression than the single-dimensional pitch value which can be very noisy. This feature has seven dimensions in the default configuration given by the Janus Recognition Toolkit.

Other features we tried include the Mel-frequency cepstral coefficients (MFCC) with 20 dimensions [ref] and a set of bottleneck features trained on phoneme recognition using a feed forward network, which is used for speech recognition at the Interactive Systems Lab [ref].

Because our training data set is limited, we easily run into overfitting problems with a large input feature dimension.

All of the features are extracted with a window size of 32 milliseconds, overlapping each other with a stride of 10 milliseconds. This gives us 100 frames per second.

2.4 Training and Neural Network Design

We begin with a simple feed forward network. The input layer consists of all the chosen features over a fixed time context. With a time context of $c\,\mathrm{ms}$ and a feature dimension of f, this gives us a input dimension of $f \times \lfloor \frac{c}{10\,\mathrm{ms}} \rfloor$. One or more hidden layers with varying numbers of neurons follow. After every layer we apply an activation or nonlinear function like tanh or ReLU. The output layer is $(n+1)$–dimensional, where n is the number of backchannel categories we want to predict. This layer has softmax as an activation function, which maps a K-dimensional vector of arbitrary values to values that are in the range $(0,1]$ and that add up to 1, which allows us to interpret them as class probabilities.

We then calculate categorical cross-entropy of the output values of the network and the ground truth from the training data set. This gives us the loss function as the function mapping from the network inputs to the cross-entropy output. We can now train the parameters of the network by deriving it individually for each of the neurons and descending the resulting gradient using the back-propagation algorithm (Rumelhart, Hinton, and Williams 1986).

In the simplest case (ignoring different kinds of backchannels) we train the network on the outputs $[1,0]$ for backchannels and $[0,1]$ for non-backchannels. A visualization of this architecture can be seen in fig. 2.1. In the following sections we will concentrate on this architecture.

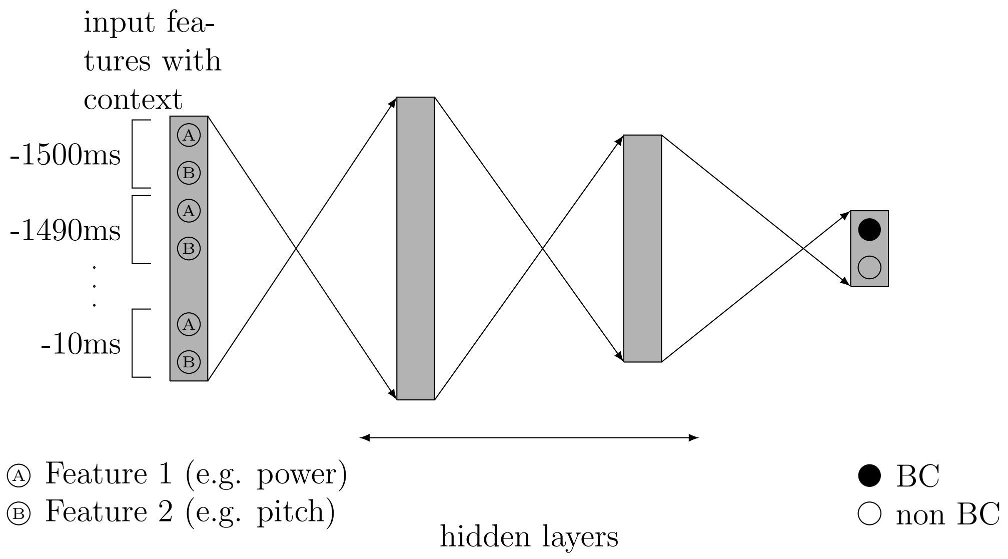


Figure 2.1: Neural Network architecture.

The placement of backchannels is dependent on previous backchannels: If the previous backchannel utterance was a long time ago, the probability of a backchannel happening shortly is higher and vice versa. To accommodate for this, we want the network to also take its previous internal state or outputs into account. We do this by modifying the above architecture to use Long-short term memory layers (LSTM) instead of feed forward layers. LSTM neurons are recurrent, meaning they are connected to themselves in a time-delayed fashion, and they have an internal state cell which is transmitted through time and which has set and clear functions which are triggered by any combination of their inputs. LSTM networks are trained in similar fashion as feed forward networks, with the time-stacked layer instances unrolled into individual copies with shared parameters before applying the backpropagation algorithm.

2.5 Postprocessing

Our goal is to generate an artificial audio track containing utterances such as “uh-huh” or “yeah” at appropriate times. The neural neural network gives us a noisy value between 0 and 1, which we interpret as the probability of a backchannel happening at the given time. To generate our audio track from this output, we need to convert the noisy floating value into discrete trigger timestamps. We first run a low-pass filter over the network output, which gives us a less noisy and more continous output function. Then we use a fixed trigger threshold to extract the ranges in the output where the predictor is fairly confident that a backchannel should happen.

Within these ranges we have multiple possibilities to choose the trigger anchor, which is either the same as the actual backchannel trigger, or a fixed time before it.

The easiest method is to use the time of the maximum peak of every range where the value is larger than the threshold, but this requires us to wait until the value is lower than the threshold again before we can decide where the maximum is, which introduces another delay and is thus bad for live detection.

Another possibility is to use the start of the range, but this can give us worse results because it might force the trigger to happen earlier than the time the network would give the highest rating.

A compromise between the best quality and immediate decision is to use the first local maximum within the thresholded range. Because of the low-pass filter we mostly have few local maxima which differ from the global maximum within the given range, and it’s easy to decide when the local maximum was reached by simply triggering as soon as the first derivate is < 0.

An example of this postprocessing procedure can be seen in fig. 2.2.

Another problem that arises is which low-pass filter to use. A simple gaussian blur uses future information which we do not want. One approach is to cut off the gaussian filter at some multiple of the standard deviation and have it offset to the left so the newest frame it uses is the present frame. Of course this causes the prediction to be delayed further. Another method is to use a predictive filter such as a Kalman Filter (Kalman 1960) instead.

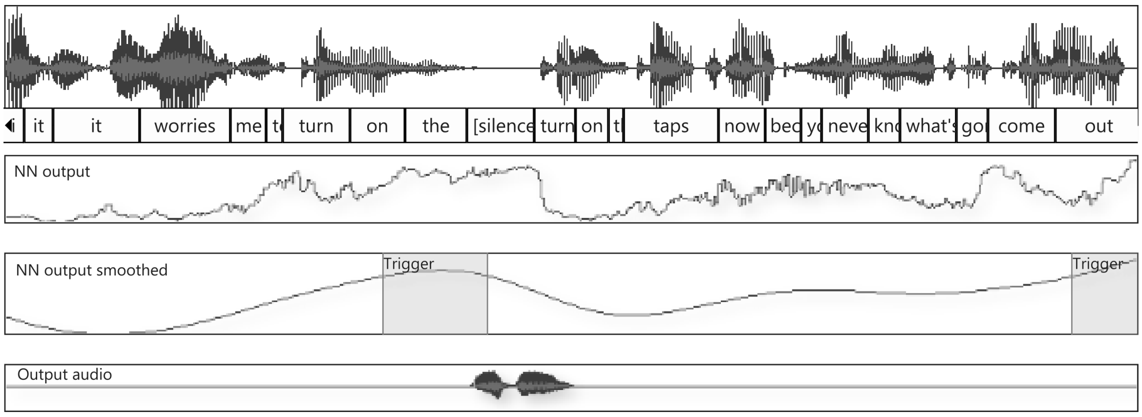


Figure 2.2: Postprocessing example

2.6 Evaluation

Because our predictor is only capable of handling situations where one speaker is consistently speaking and the other consistently listening, we need to evaluate it on only those segments. We call these segments "monologuing segments".

For a simple subjective evaluation, we take a some random audio tracks from the training data and extract the monologuing segments. For each segment, we remove the original listener channel and replace it with the artificial one. This audio data is generated by inserting a random backchannel audio sample at every predicted timestamp. We get these audio samples from a random speaker from the training data, keeping the speaker the same over the whole segment so it sounds like a specific person is listening to the speaker.

To get an objective evaluation of the performance of our predictions, we again take a set of monologuing segments from the evaluation data set and compare the prediction with the ground truth, i.e. all the timestamps where a backchannel happens in the original data.

We interpret a prediction as correct if it is within a specific margin of the nearest backchannel in the dataset. For example, with a margin of error of $[-100\,\mathrm{ms}, 300\,\mathrm{ms}]$, if the real data has a backchannel at 5.5 seconds, we say the predictor is correct if it also produces a backchannel within $[5.5\,\mathrm{s} - 100\,\mathrm{ms}, 5.5\,\mathrm{s} + 300\,\mathrm{ms}] = [5.4\,\mathrm{s}, 5.8\,\mathrm{s}]$.

In other research, varying margins of error have been used. We use a margin of 0ms to +1000ms for our initial tests, and later also do our evaluation with other margins for comparison with related research.

After aligning the prediction and the ground truth with the margin of error, we get two overlapping sets of timestamps. The set of predictions is called "selected", the set of true elements is called "relevant". The number of true positives is defined as $TP = |selected \cap relevant|$. The number of false positives is defined as $FP = |selected \setminus relevant|$. The number of false negatives is defined as $FN = |relevant \setminus selected|$.

With these, we can now calculate the measures *Precision* and *Recall* commonly used in information retrieval and binary classification:

$$\begin{aligned} \mathit{Precision} &= \frac{TP}{TP + FP} \\ \mathit{Recall} &= \frac{TP}{TP + FN} \end{aligned}$$

Both of these are values between 0 and 1. The *Precision* value is the fraction of returned values that were correct, and can thus be seen as a measure of the *quality* of a algorithm. The *Recall* value, also known as *sensitivity* is the fraction of correct values that the algorithm output, thus it can be interpreted as a measure of *quantity*. Precision and Recall are in a inverse relationship, and it is usually possible to increase one of them while reducing the other by tweaking parameters of the algorithm. Recall can be easily maximized to 100% by simply returning true for every given timestamp. Precision can be maximized by never outputting anything, causing every predicted value to be correct. To solve this problem, we use the normalized harmonic mean of precision and recall, also known as the F1-Score or F-Measure:

$$\mathit{F1\ Score} = 2 \cdot \frac{1}{\frac{1}{\mathit{Recall}} + \frac{1}{\mathit{Precision}}} = 2 \cdot \frac{\mathit{Precision} \cdot \mathit{Recall}}{\mathit{Precision} + \mathit{Recall}}$$

We use the F1-Score to objectively measure the performance of our predictors.

3. Experimental Setup

3.1 Dataset

We used the switchboard dataset (Godfrey and Holliman 1993) which consists of 2438 telephone conversations of five to ten minutes, 260 hours in total. These telephone conversations have complete transcriptions and word alignments (Harkins and others 2003). As opposed to many other data sets, the transcriptions also contain backchannel utterances like *uh-huh* and *yeah*, making it ideal for our task.

The transcriptions are split into *utterances*, which are multiple words grouped by speech structure, for example: "did you want me to go ahead / okay well one thing i- i- i guess both of us have very much aware of the equality / uh it seems like women are uh just starting to really get some kind of equality not only in uh jobs but in the home where husbands are starting to help out a lot more than they ever did um". The slashes indicate utterance boundaries. Each of these utterance has a start time and stop time attached, where the stop time of one utterance is always the same as the start time of the next utterance. For longer periods of silence, a "[silence]" utterance is between them.

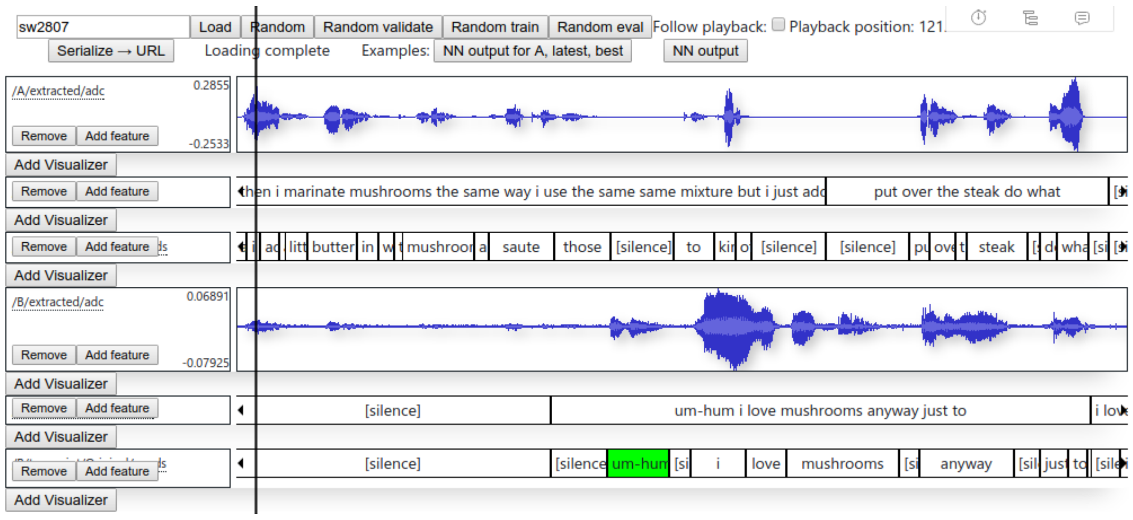
The word alignments have the same format, except they are split into single words, each with start and stop time, with silence utterances being far more frequent.

To better understand and visualize the dataset, we first wrote a complete visualization GUI for viewing and listening to audio data, together with transcriptions, markers and other data. This proved to be amazingly helpful. A screenshot of the UI inspecting a short portion of one of the phone conversations can be seen in fig. 3.1.

3.2 Extraction

3.2.1 Backchannel utterance selection

We used annotations from The Switchboard Dialog Act Corpus (Jurafsky, Van Ess-Dykema, and others 1997) to decide which utterances to classify as backchannels. The SwDA contains categorical annotations for utterances for about half of the data of the Switchboard corpus. An excerpt of the most common categories can be seen in tbl. 3.1.



then i marinate mushrooms the same way i use the same same mixture but i just add | put over the steak do what

[silence] | um-hum i love mushrooms anyway just to | i love

[silence] | [silence | um-hum | [si | i | love | mushrooms | [si | anyway | [sil | just | to | [sile

Figure 3.1: From top to bottom: Speaker A audio data, transcription, and word alignement, then the same for speaker B.

Table 3.1: Most common categories from the SwDA Corpus

	name	act_tag	example	train_count	full_count
1	Statement-non-opinion	sd	Me, I'm in the legal department.	72824	75145
2	Acknowledge (Backchannel)	b	Uh-huh.	37096	38298
3	Statement-opinion	sv	I think it's great	25197	26428
4	Agree/Accept	aa	That's exactly it.	10820	11133
5	Abandoned or Turn-Exit	%	So, -	10569	15550
6	Appreciation	ba	I can imagine.	4633	4765
7	Yes-No-Question	qy	Do you have to have any special training?	4624	4727

We extracted all utterances containing one of the tags beginning with "b", and counted their frequency.

We chose to use the top 150 unique utterances marked as backchannels from this set. For the most common ones, see tbl. 3.2.

Table 3.2: Most common backchannel utterances in the SwDA dataset

aggregated	self	count	category	text
31.92%	31.92%	14319	b	uh-huh
61.08%	29.15%	13075	b	yeah
68.95%	7.87%	3532	b	right
71.73%	2.78%	1249	b	oh
73.69%	1.96%	877	b	[silence]
75.09%	1.40%	629	b	oh yeah
76.49%	1.40%	627	b	yes
77.84%	1.35%	607	b	okay
78.86%	1.02%	458	bk	okay
79.75%	0.89%	399	b	huh
80.57%	0.81%	364	b	sure

identify utterances as backchannels just by their text. As can be seen in tbl. 3.2, the SwDA also has some silence utterances marked as backchannels, which we can't distinguish from normal silence. We manually removed some requests for repetition like "excuse me" from the SwDA list, and added some other utterances that were missing from the SwDA transcriptions but present in the original transcriptions, by going through the most common utterances and manually selecting those that seemed relevant such as 'um-hum yeah', 'absolutely', 'right uh-huh'.

In total we now have a list of 161 backchannel utterances. The most common backchannels in the data set are "yeah", "um-hum", "uh-huh" and "right", adding up to 41860 instances or 68% of all extracted backchannel phrases.

The transcriptions also contain markers indicating laughter while talking (e.g. "i didn't think that well we wouldn't still be [laughter-living] [laughter-here] so …"), laughter on its own (`[laughter]`), noise markers (`[noise]`) and markers for different pronunciations (for example "mhh-kay" is transcribed as `okay_1`). To select which utterances should be categorized as backchannels and used for training, we first filter noise and other markers from the transcriptions, for example `[laughter-yeah] okay_1 [noise]` becomes `yeah okay` and then compare the resulting text to our list of backchannel phrases.

Some utterances such as "uh" can be both backchannels and speech disfluencies. For example: "… pressed a couple of the buttons up in the / uh / the air conditioning panel i think and uh and it reads out codes that way". Note that the first *uh* is it's own utterance and would thus be seen by our extractor as a backchannel. The second utterance has normal speech around it so we would ignore it. We only want those utterances that are actual backchannels, so after filtering by utterance text we only choose those that have either silence or another backchannel before them.

This gives us the following selection algorithm:

```
def is_backchannel(utterance):
    text = noise_filter(utterance)
    if index(utterance) == 0: return False
    previous_text = noise_filter(previous(utterance))
    return (text in valid_backchannels and
            (is_silent(previous_text) or is_backchannel(previous(utterance)))
```

This method gives us a total of 61645 backchannels out of 391593 utterances (15.7%) or 71207 out of 3228128 words (2.21%).

3.2.2 Feature extraction

3.2.2.1 Prosodic features

We used the Janus Recognition Toolkit (Levin et al. 2000) for parts of the feature extraction (power, pitch tracking, FFV, MFCC). These features are extracted for 32ms frame windows, with a shift of 10ms. A sample of the pitch and power features can be seen in fig. 3.2.

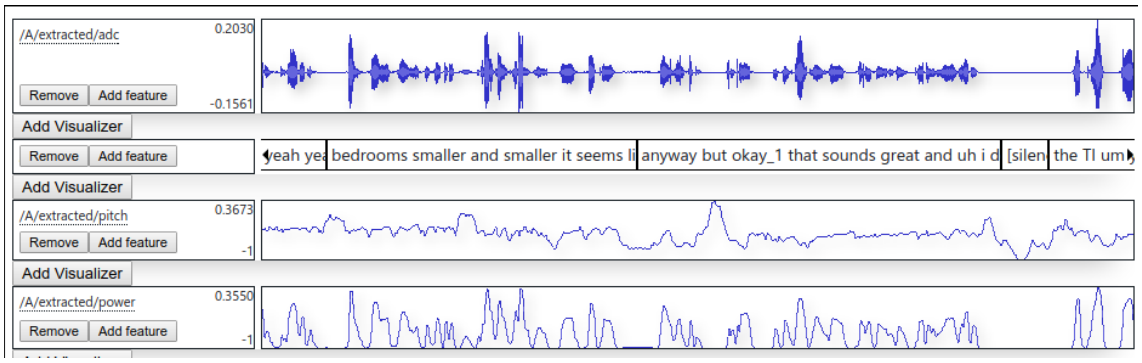


Figure 3.2: Audio samples, transcription, pitch and power for a single audio channel. Note that the pitch value is only meaningful when the person is speaking.

3.2.2.2 Linguistic features

In addition to these prosodic features, we also tried training Word2Vec (Mikolov et al. 2013) on the Switchboard dataset. Word2Vec is a "Efficient Estimation of Word Representations in Vector Space". After training it on a lot of text, it will learn the meaning of the words from the contexts they appear in, and then give a mapping from each word in the vocabulary to a n-dimensional vector, where n is configurable. Similar words will appear close to each other in this vector space, and it's even possible to run semantic calculations on the result. For example calculating $king - man + woman$ gives the result $= queen$. Because our dataset is fairly small, we used relatively small word vectors (5 - 20 dimensions).

For simplicity, we extract these features parallel to those output by Janus, with a 10 millisecond frame shift. To ensure we don't use any future information, we extract the word vector for the last word that ended *before* the current frame timestamp. This way the predictor is in theory still online, though this assumes the existence of a speech recognizer with instant output.

3.2.2.3 Context and stride

We extract the features for a fixed time context. Then we use a subset of that range as the area we feed into the network. As an example, we can extract the range [-2000ms, 0ms] for every feature, giving us 200 frames. We train the network on 1500ms of context, so we treat every offset like [-2000, -500ms], [-1990ms, -490ms], . . . , [-1500ms, 0ms] as individual training samples. This gives us 50 training samples per backchannel utterance, greatly increasing the amount of training data, but introducing more smear as the network needs to learn to handle a larger variance in when the backchannel cue appears in its inputs, and thus reducing the confidence of its output.

This turned out to not work very well, so in the end we settled on only extracting the features for the range [-w - 10ms, 0] where w is the context width, and training the network on [-w - 10ms, 10ms] and [-w, 0ms]. This gives us two training samples per utterance, reduces the smearing problem and at the same time force the network to learn to correctly handle when its inputs are the same or similar but offset by one.

We can also choose to only use every n-th timestep, which we call the "context stride". This works under the assumption that the input features don't change with a high frequency, which greatly reduces the input dimension and speeds up training. In practice a stride of 2 worked well, meaning one frame every 20 milliseconds. This works great in combination with the above. For example, with a stride of 2 and a context size of 100ms, we would now get these two training samples (described as frame indices relative to the onset of the backchannel utterance):

1. [-10, -8, -6, -4, -2, 0]
2. [-11, -9, -7, -5, -3, -1]

3.3 Training

We used Theano (Theano Development Team 2016) with Lasagne v1.0-dev (Dieleman et al. 2015) for rapid prototyping and testing of different parameters. We trained a total of over 200 network configuration with various context lengths (500ms to 2000ms), context strides (1 to 4 frames), network depths ranging from one to four hidden layers, layer sizes ranging from 15 to 100 neurons, activation functions (tanh and relu), gradient descent methods (SGD, Adadelta and Adam), dropout layers (0 to 50%) and layer types (feed forward and LSTM).

In general, we used three variables to monitor training: Training loss, which is what the network optimizes, as defined in sec. 2.4. Validation loss, which is the same function, but on the seperate validation data set, and the validation error, which we define as

$$1 - \frac{\sum_{s \in S}\{1 \text{ if prediction}(s) = \text{truth}(s) \text{ else } 0\}}{|S|},$$

where S is all the frames for all the samples in the validation data set.

We started with a simple model with a configuration of pitch and power as input and 800 ms of context, giving us $80 \cdot 2 = 160$ input dimensions, hidden layers of 100 $\rightarrow$ 50 feed forward neurons. We trained this using many different gradient descent methods such as stochastic gradient descent (SGD), SGD with momentum, Nesterov momentum, Adadelta and Adam, each with fixed learning rates to start. The momentum methods add a speed variable to the descent. This can be interpreted similar to its physical name giver. Imagine a ball rolling down a mountain slope. For each time period, it keeps it's previous momentum and is thus able to jump over small dents in the ground (local minima). In our case, momentum worsened the results, so we stayed with SGD and Adadelta.

We tried different weight initialization methods. Initializing all weights with zero gave significantly worse results than random initialization, so we stayed with the Lasagne default of Glorot uniform initialization, which uses uniformly distributed random values, with the maximum value scaled so it makes statistical sense in the layer dimensions (Glorot and Bengio 2010).

We compared online and offline prediction, where offline prediction got 400 ms of past audio and 400 ms of future audio from the onset of the backchannel utterance. Offline prediction gave 18% better results, but of course we are more interested in online prediction.

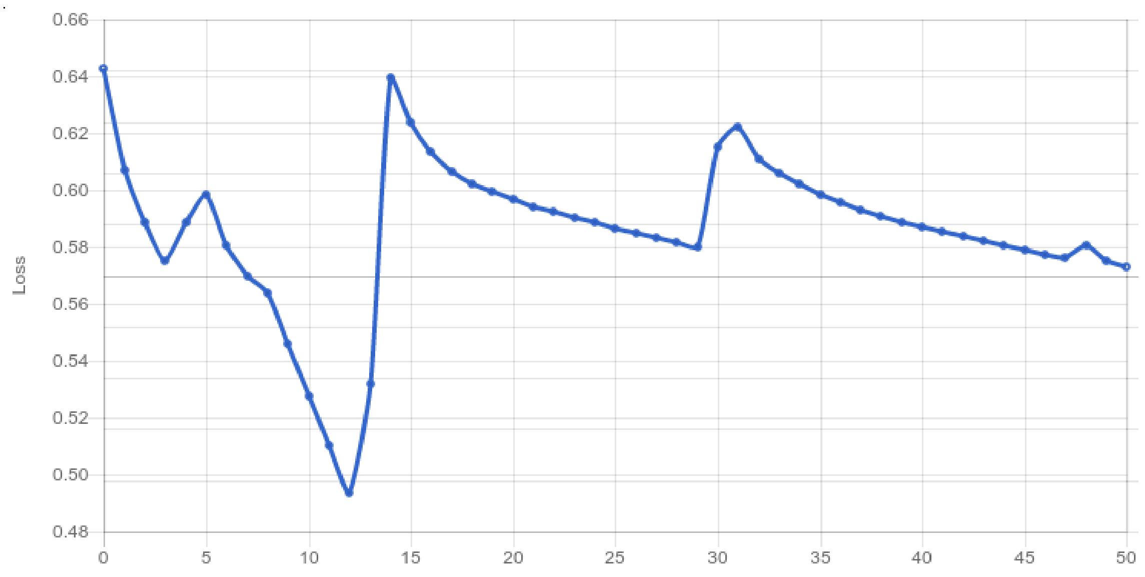


Figure 3.3: Exploding gradient while training a LSTM network. Shown is the training loss over epochs

The first simple LSTM we tested by simply replacing the feed forward layers with LSTM layers immediately improved the results by 16% without any further adjustments. But this showed the issues with a fixed learning rate, as the gradient regularily exploded after every 10 - 15 epochs, as can be seen in fig. 3.3. One epoch is defined as one whole backward pass of all the training data through the network. When adding FFV, increasing the input dimension per time frame from 2 to 9, SGD stopped working at all without manually tuning the learning rate.

These problem was solved by using Adam (Kingma and Ba 2014) instead of SGD, which is a gradient descent method related to Adadelta and Adagrad which also incorporates momentum in some way, see fig. 3.4. No one really understands how these work, but Adam with a fixed learning rate of 0.001 worked great for us, so we did all further testing using Adam.

The LSTM networks we tested were prone to overfitting very quickly, but they still provided better results after two to three epochs than normal feed forward networks after 100 epochs. Overfitting happens when the results still improve on the training data set, but plateau or get worse on the validation data set. This means the network is starting to learn specific quirks in the training data set by heart, which it then can't apply on other data.

We tried adding dropout layers to the networks to try and avoid overfitting and to generally improve the results. Dropout layers randomly disconnect a specified fraction of neurons in a layer, different for every training batch. This should in theory help the network interpret it's inputs even when it is partially "blind". For validation the dropout is deactivated, so the network is able to take advantage of every single feature when actually using it as a predictor. In this case, we tried adding different dropout settings such as "input (20% dropout) $\rightarrow$ 125 neurons (50% dropout) $\rightarrow$ 80 neurons (50% dropout) $\rightarrow$ output" but this only increased the noise in the training loss and did not improve the results over a simple "input $\rightarrow$ 70 neurons $\rightarrow$ 45 neurons" configuration, both for feed forward and for LSTM networks.

```
**Require:** $\alpha$: Stepsize
**Require:** $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
**Require:** $f(\theta)$: Stochastic objective function with parameters $\theta$
**Require:** $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$
  $v_0 \leftarrow 0$
  $t \leftarrow 0$
  **while** $\theta_t$ not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$
  **end while**
  **return** $\theta_t$ (Resulting parameters)
```

Figure 3.4: The Adam method for stochastic optimization. I have no idea what any of those letters mean, but it's been cited a bazillion times and works great.

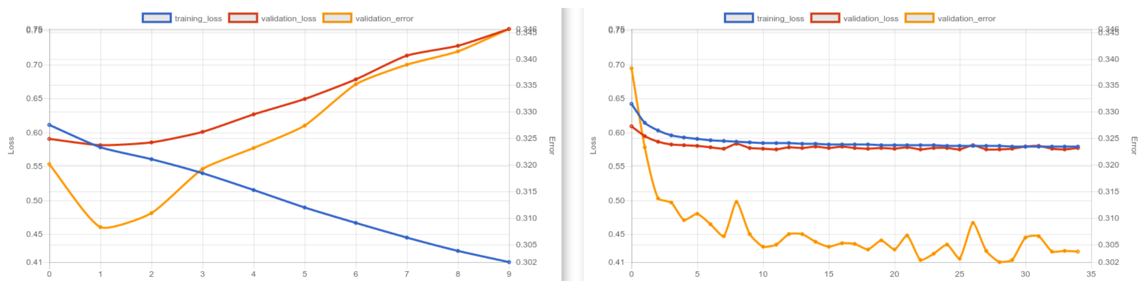


Figure 3.5: The same LSTM network trained without (left) and with (right) L2-Regularization. Note that without regularization the network starts overfitting after two epochs. With regularization training and validation loss mostly stay the same with regularization, and the validation error continues to improve. Training loss is blue, validation loss is red and validation error is orange.

L2-Regularization reduced this problem and slightly improved the results, as can be seen in the example in fig. 3.5.

3.4 Evaluation

The training data contains two-sided conversations. Because the output of our predictors is only relevant for segments with only one person talking, we only run our evaluation on monologuing segments.

For this we define the predicate `is_listening` (is only emitting utterances that are backchannels or silence) and `is_talking` (`= not is_listening`). A monologuing segment is the maximum possible time range in which one person is consistently talking and the other only listening. Additionally, we only consider such segments of a minimum length of 5 seconds to reduce problems arising from predictions at the edges of the segments, though this only improved the resulting F1-scores by 2%.

4. Results

All the results in Figure 4.1 use the following configuration if not otherwise stated: LSTM, Hidden layers: 70 $\rightarrow$ 35 neurons, Input features: Power, pitch, FFV, Context frame stride: 2, Margin of Error: 0ms to +1000ms. Precision, Recall and F1-Score are given for the validation data set. In Figure 4.2, our final results are given for the completely independent evaluation data set.

4.1 Conclusion

Figure 4.1: Results on the Validation Set

(a) Results with various context lengths

Context	Precision	Recall	F1-Score
500ms	0.219	0.466	0.298
1000ms	0.280	0.497	0.358
1500ms	0.305	0.488	**0.375**
2000ms	0.275	0.577	0.373

(b) Results with various network configurations

Layers	Precision	Recall	F1-Sc
$in \rightarrow 100 \rightarrow out$	0.280	0.542	0.36
$in \rightarrow 50 \rightarrow 20 \rightarrow out$	0.291	0.506	0.37
$in \rightarrow 70 \rightarrow 35 \rightarrow out$	0.305	0.488	**0.37**
$in \rightarrow 100 \rightarrow 50 \rightarrow out$	0.303	0.473	0.36
$in \rightarrow 70 \rightarrow 50 \rightarrow 35 \rightarrow out$	0.278	0.541	0.36

(c) Results with various input features.

Features	Precision	Recall	F1-Score
power, ffv	0.259	0.513	0.344
power, pitch	0.307	0.435	0.360
power, pitch, mfcc	0.278	0.514	0.360
power, ffv, mfcc	0.279	0.515	0.362
power, pitch, word2vec$_{dim=5}$	0.304	0.474	0.370
power, pitch, ffv, word2vec$_{dim=5}$	0.297	0.501	0.373
power, pitch, ffv	0.305	0.488	0.375
power, pitch, word2vec$_{dim=10}$	0.316	0.479	**0.381**

(d) Results with various context fram

Stride	Precision	Recall	F1
1	0.290	0.490	(
2	0.305	0.488	**0**
4	0.285	0.498	(

(e) Feed forward vs LSTM

Layers	Precision	Recall	F1-Score
Feed Forward ($in \rightarrow 100 \rightarrow 50 \rightarrow out$)	0.242	0.490	0.324
Feed Forward ($in \rightarrow 70 \rightarrow 35 \rightarrow out$)	0.251	0.468	0.327
LSTM ($in \rightarrow 70 \rightarrow 35 \rightarrow out$)	0.305	0.488	**0.375**

Figure 4.2: Final best results on the evaluation set (chosen by validation set)

(a) Comparison with previous research. Mueller et al. did their evaluation without the constraints defined in section 3.4, so we adjusted our baseline and evaluation to match their setup

Predictor	Precision	Recall	F1-Score
Baseline (random)	0.0417	0.0417	0.0417
Müller et al. (offline) [?]	–	–	0.109
Our results (online)	0.0929	0.378	**0.149**

(b) Results with various margins of error used in other research [?]

Margin of Error	Constraint	Precision	Recall	F1-Score
−200 ms to 200 ms		0.163	0.348	0.222
−100 ms to 500 ms		0.225	0.414	0.291
−500 ms to 500 ms		0.247	0.536	0.339
0 ms to 1000 ms	Baseline (correct BC count at random times)	0.111	0.0521	0.0708
	Balanced Precision and Recall	**0.331**	0.329	0.330
	Best F1-Score (only prosodic features)	0.294	0.488	**0.367**
	Best F1-Score (including word2vec)	0.300	0.473	**0.367**

5. Conclusion

Bibliography

Cathcart, Nicola, Jean Carletta, and Ewan Klein. 2003. "A Shallow Model of Backchannel Continuers in Spoken Dialogue." In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 1*, 51–58. EACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics. doi:10.3115/1067807.1067816.

De Kok, Iwan, and Dirk Heylen. 2009. "Multimodal End-of-Turn Prediction in Multi-Party Meetings." In *Proceedings of the 2009 International Conference on Multimodal Interfaces*, 91–98. ACM. http://dl.acm.org/citation.cfm?id=1647332.

de Kok, Iwan, Dirk Heylen, and Louis-Philippe Morency. 2013. "Speaker-Adaptive Multimodal Prediction Model for Listener Responses." In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, 51–58. ICMI '13. New York, NY, USA: ACM. doi:10.1145/2522848.2522866.

de Kok, Iwan, Derya Ozkan, Dirk Heylen, and Louis-Philippe Morency. 2010. "Learning and Evaluating Response Prediction Models Using Parallel Listener Consensus." In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, 3:1–3:8. ICMI-Mlmi '10. New York, NY, USA: ACM. doi:10.1145/1891903.1891908.

de Kok, Iwan, Ronald Poppe, and Dirk Heylen. 2014. "Iterative Perceptual Learning for Social Behavior Synthesis." *Journal on Multimodal User Interfaces* 8 (3): 231–41. doi:10.1007/s12193-013-0132-1.

Dieleman, Sander, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, and others. 2015. "Lasagne: First Release." August. doi:10.5281/zenodo.27878.

Fujie, Shinya, Kenta Fukushima, and Tetsunori Kobayashi. 2004. "A Conversation Robot with Back-Channel Feedback Function Based on Linguistic and Nonlinguistic Information." In *Proc. ICARA Int. Conference on Autonomous Robots and Agents*, 379–84.

Glorot, Xavier, and Yoshua Bengio. 2010. "Understanding the Difficulty of Training Deep Feedforward Neural Networks." In *In Proceedings of the International Con-*

ference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics.

Godfrey, John, and Edward Holliman. 1993. "Switchboard-1 Release 2." https://catalog.ldc.upenn.edu/ldc97s62.

Harkins, Dan, and others. 2003. "ISIP Switchboard Word Alignments." https://www.isip.piconepress.com/projects/switchboard/.

Huang, Lixing, Louis-Philippe Morency, and Jonathan Gratch. 2010. "Learning Backchannel Prediction Model from Parasocial Consensus Sampling: A Subjective Evaluation." In *Intelligent Virtual Agents*, 159–72. Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-15892-6_17.

———. 2011. "Virtual Rapport 2.0." In *Intelligent Virtual Agents*, edited by Hannes Högni Vilhjálmsson, Stefan Kopp, Stacy Marsella, and Kristinn R. Thórisson, 68–79. Lecture Notes in Computer Science. Springer Berlin Heidelberg. doi:10.1007/978-3-642-23974-8_8.

Jurafsky, Daniel, Carol Van Ess-Dykema, and others. 1997. "Switchboard Discourse Language Modeling Project."

Kalman, Rudolph Emil. 1960. "A New Approach to Linear Filtering and Prediction Problems." *Transactions of the ASME–Journal of Basic Engineering* 82 (Series D): 35–45.

Kingma, Diederik P., and Jimmy Ba. 2014. "Adam: A Method for Stochastic Optimization," December. http://arxiv.org/abs/1412.6980.

Kitaoka, N., M. Takeuchi, Nishimura R, and Nakagawa S. 2005. "Response Timing Detection Using Prosodic and Linguistic Information for Human-Friendly Spoken Dialog Systems" 20 (3): 220–28. doi:10.1527/tjsai.20.220.

Laskowski, Kornel, Mattias Heldner, and Jens Edlund. 2008. "The Fundamental Frequency Variation Spectrum." *Proceedings of FONETIK 2008*, 29–32.

Levin, Lori, Alon Lavie, Monika Woszczyna, Donna Gates, Marsal Gavaldá, Detlef Koll, and Alex Waibel. 2000. "The Janus-III Translation System: Speech-to-Speech Translation in Multiple Domains." *Machine Translation* 15 (1): 3–25. doi:10.1023/A:1011186420821.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. "Efficient Estimation of Word Representations in Vector Space," January. http://arxiv.org/abs/1301.3781.

Morency, Louis-Philippe, Iwan de Kok, and Jonathan Gratch. 2010. "A Probabilistic Multimodal Approach for Predicting Listener Backchannels." *Autonomous Agents and Multi-Agent Systems* 20 (1): 70–84. doi:10.1007/s10458-009-9092-y.

Morency, Louis-Philippe, Iwan de Kok, and Jonathan Gratch. 2008. "Predicting Listener Backchannels: A Probabilistic Multimodal Approach." In *Intelligent Virtual Agents*, 176–90. Springer, Berlin, Heidelberg. doi:10.1007/978-3-540-85483-8_18.

Mueller, Markus, David Leuschner, Lars Briem, Maria Schmidt, Kevin Kilgour, Sebastian Stueker, and Alex Waibel. 2015. "Using Neural Networks for Data-Driven Backchannel Prediction: A Survey on Input Features and Training Techniques." In

Human-Computer Interaction: Interaction Technologies, 329–40. Springer, Cham. doi:10.1007/978-3-319-20916-6_31.

Nishimura, Ryota, Norihide Kitaoka, and Seiichi Nakagawa. 2007. "A Spoken Dialog System for Chat-Like Conversations Considering Response Timing." In *Text, Speech and Dialogue*, 599–606. Springer, Berlin, Heidelberg. doi:10.1007/978-3-540-74628-7_77.

Noguchi, Hiroaki, and Yasuharu Den. 1998. "Prosody-Based Detection of the Context of Backchannel Responses." In *ICSLP*.

Okato, Y., K. Kato, M. Kamamoto, and S. Itahashi. 1996. "Insertion of Interjectory Response Based on Prosodic Information." In*, Third IEEE Workshop on Interactive Voice Technology for Telecommunications Applications, 1996. Proceedings*, 85–88. doi:10.1109/IVTTA.1996.552766.

Ozkan, D., and L. P. Morency. 2013. "Latent Mixture of Discriminative Experts." *IEEE Transactions on Multimedia* 15 (2): 326–38. doi:10.1109/TMM.2012.2229263.

Ozkan, Derya, and Louis-Philippe Morency. 2010. "Concensus of Self-Features for Nonverbal Behavior Analysis." In *Human Behavior Understanding*, 75–86. Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-14715-9_8.

Ozkan, Derya, Kenji Sagae, and Louis-Philippe Morency. 2010. "Latent Mixture of Discriminative Experts for Multimodal Prediction Modeling." In *Proceedings of the 23rd International Conference on Computational Linguistics*, 860–68. COLING '10. Stroudsburg, PA, USA: Association for Computational Linguistics. http://dl.acm.org/citation.cfm?id=1873781.1873878.

Poppe, Ronald, Khiet P. Truong, Dennis Reidsma, and Dirk Heylen. 2010. "Backchannel Strategies for Artificial Listeners." In *Intelligent Virtual Agents*, 146–58. Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-15892-6_16.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1986. "Learning Representations by Back-Propagating Errors." *Nature* 323 (6088): 533–36. doi:10.1038/323533a0.

Takeuchi, Masashi, Norihide Kitaoka, and Seiichi Nakagawa. 2004. "Timing Detection for Realtime Dialog Systems Using Prosodic and Linguistic Information." In *Speech Prosody 2004, International Conference*.

Theano Development Team. 2016. "Theano: A Python Framework for Fast Computation of Mathematical Expressions." *arXiv E-Prints* abs/1605.02688 (May). http://arxiv.org/abs/1605.02688.

Truong, Khiet P., R. W. Poppe, and D. K. J. Heylen. 2010. "A Rule-Based Backchannel Prediction Model Using Pitch and Pause Information." http://eprints.eemcs.utwente.nl/18627/.

Ward, Nigel. 1996. "Using Prosodic Clues to Decide When to Produce Back-Channel Utterances." In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth*

International Conference on, 3:1728–31. IEEE. http://ieeexplore.ieee.org/abstract/document/607961/.

Ward, Nigel, and Wataru Tsukahara. 2000. “Prosodic Features Which Cue Back-Channel Responses in English and Japanese.” *Journal of Pragmatics* 32 (8): 1177–1207.

Watanabe, Tmio, and Naohiko Yuuki. 1989. “A Voice Reaction System with a Visualized Response Equivalent to Nodding.” In *Proceedings of the Third International Conference on Human-Computer Interaction, Vol.1 on Work with Computers: Organizational, Management, Stress and Health Aspects*, 396–403. New York, NY, USA: Elsevier Science Inc. http://dl.acm.org/citation.cfm?id=92158.92234.