# Dialogue Breakdown Detection in Chatbots


**student**: Mariya Hendriksen
**supervisor**: Prof. Dr. Marie-Francine Moens
**assessor**: Prof. Dr. Hugo Van Hamme
**mentor**: Artuur Leeuwenberg

Faculty of Engineering Science


February 1, 2019

# Outline


Introduction

Data & Methods
- Dataset
- Model Type & Architecture
- Word Embedding Models

Implementation
- Data Preprocessing
- Models Development

Evaluation
- Evaluation Metrics
- Evaluation Results

Conclusion

# Outline

Introduction

Data & Methods
- Dataset
- Model Type & Architecture
- Word Embedding Models

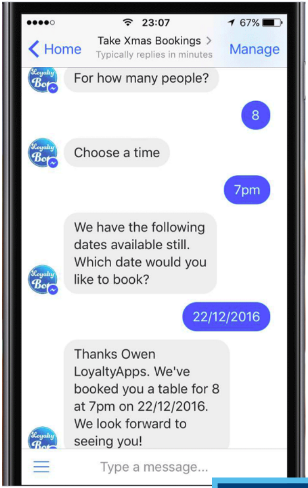
Implementation
- Data Preprocessing
- Models Development

Evaluation
- Evaluation Metrics
- Evaluation Results

Conclusion

# Chatbot

- A chatbot is a dialogue system which conducts a conversation with a user via text or speech.
- The popularity of chatbots increase.
- Instant 24 hours service
- Usage categories: website help, education, customer service, e-commerce, entertainment, etc.






KU LEUVEN

# Dialogue Breakdown

- Breakdown is a point in a dialogue when the interaction is interrupted with or without completion of the performed task [Martinovsky and Traum, 2006]

It's nice to *go shopping alone*.
I agree. That's nice.
Shopping takes time.
Window shopping is also fun.
It's fun to *go shopping with somebody*.

Chatbot utterances are red.

# About the Project

- Aim: create a dialogue breakdown detection model
- Research questions:
  - Before building the model
    - What models for breakdown detection are available?
    - Which types of models perform better?
  - After building the model
    - How does the architecture of the model influence its performance?
    - How does the embedding type of the detector influence its performance?

# Contributions

- Overview and comparison of the existing dialogue breakdown detectors
- Introduction of dialogue breakdown detection models which outperform the baseline
- Exploration of relationship between model architecture and its performance
- Investigation of the relationship between embedding type and detector's performance

# Related Literature

- Dialogue breakdown detection challenge:
  - introduced in [Ryuichiro et al., 2015]
  - three challenges held [Higashinaka et al., 2016, 2017]
  - fourth challenge will be held during IWSDS 2019 [dbd]
- Applications:
  - re-ranking system responses [Inaba and Takahashi]
  - aggregating utterances that can be used as system responses [Sugiyama, 2016]

# Outline

Introduction

Data & Methods
- Dataset
- Model Type & Architecture
- Word Embedding Models

Implementation
- Data Preprocessing
- Models Development

Evaluation
- Evaluation Metrics
- Evaluation Results

Conclusion

# Dataset

- Dataset from the Dialogue Breakdown Detection Challenge 3 [dbdc33]
- Dialogues sources:
  - chatbots IRIS, Tick-Tock, 'Yura and Idris'
  - Conversational Intelligence Challenge
- Labels: NB, PB, B
- Classes are balanced
- 30 annotators per system utterance

# Baseline Definition

Based on the overview of existing models performance, we chose two models for the baseline:

- accuracy baseline
- $F_1(B)$ and $F_1(PB+B)$ baseline

| Baseline type | Model | Score |
| --- | --- | --- |
| Accuracy | LSTM + word2vec | 0.44 |
| $F_1(B)$ | MemNN + attention | 0.36 |
| $F_1(PB+B)$ | MemNN + attention | 0.87 |

Table: Baseline Models



KU LEUVEN

# Model Type & Architecture

- Model Type: Why LSTM?
  - can process sequential data
  - handles long-term dependencies
- Selected architecture types:
  - vanilla LSTM
  - stacked LSTM
  - bidirectional LSTM

# Word Embedding Models Selection

- Limited data $\implies$ pretrained word embedding models
- For the experiments, we select three word embedding types:
  - word2vec pretrained on Google News, 300D
  - GloVe pretrained on Twitter, 200D
  - GloVe pretrained on Common Crawl, 300D

# Outline


Introduction

Data & Methods
- Dataset
- Model Type & Architecture
- Word Embedding Models

Implementation
- Data Preprocessing
- Models Development

Evaluation
- Evaluation Metrics
- Evaluation Results

Conclusion

# Datasets Preprocessing

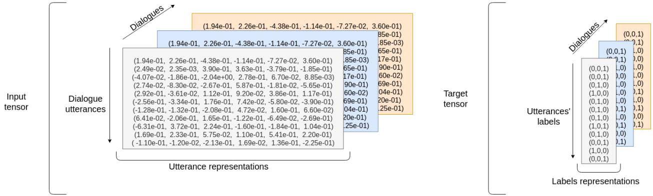
- Preprocessing steps:
  - General steps:
    - tokenization (with `nltk.tokenize.casual.TweetTokenizer`)
    - apostrophes contractions replacement, e.g., $that's \rightarrow that\ is$
  - Additional steps:
    - for word2vec pretrained on Google News: punctuation signs removal
    - for GloVe pretrained on tweets: lowercasing
- Dialogues length varies $\Longrightarrow$ padding



KU LEUVEN

# Representations

- **Token representation** obtained with pretrained embedding model
- **Utterance representation** is the average of the token embeddings comprising the utterances
- **Dialogue representation** is a 2D tensor comprising user and system utterances
- Each **label** is represented with one-hot encoding



# Models Development I

- **Software**: `Python 3.6` with libraries `NumPy`, `NLTK`, `gensim`, `matplotlib`, `sklearn`, `json`. `Tensorflow`(backend) + `Keras`
- **Layers**: one input layer, two hidden layers, one output layer
- **Loss function**: categorical cross-entropy:

$$H(y, \hat{y}) = -\sum_{i=1} y_i log(\hat{y}_i) \tag{1}$$

where $y$ - ground truth label, $\hat{y}$ - predicted label

# Models Development II

- **Mini-batching** with batch size 32
- **Optimization method**: Root Mean Square Propagation (RMSProp)
- **Maximum number of epochs**: 100
- To prevent **overfitting**:
  - Early stopping with patience 5
  - Dropout and recurrent dropout, coefficients set to 0.1



KU LEUVEN

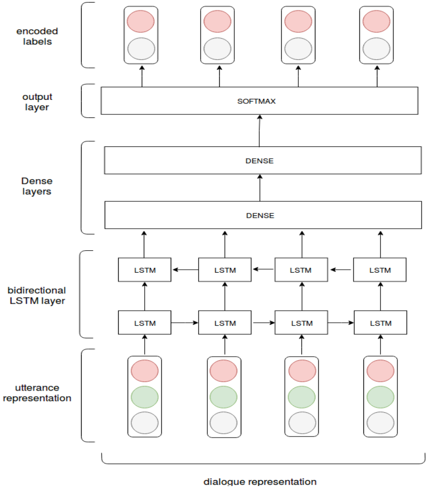
# Bidirectional LSTM

```python
# imports
from keras.models import Sequential
from keras.layers import LSTM, Dense, Bidirectional

# model definition
bi_lstm = Sequential()
# input layer
lstm1 = Bidirectional(LSTM(NUM_CELLS, return_sequences=True, dropout
↪  = DROPOUT_COEFF, recurrent_dropout = DROPOUT_COEFF),
↪  input_shape=(MAX_UTTERANCE_LENGTH, UTTERANCE_DIMENSIONALITY),
↪  merge_mode = 'sum')
bi_lstm.add(lstm1)
# hidden layers
bi_lstm.add(add(Dense(NUM_CELLS, activation = 'relu')))
bi_lstm.add(Dense(NUM_CELLS, activation = 'relu'))
# output layer
bi_lstm = Dense(NB_LABELS, activation = 'softmax')
bi_lstm.add(outputs)

# model compilation
bi_lstm.compile(
    optimizer = 'rmsprop',
    loss = 'categorical_crossentropy',
    metrics=['accuracy'])

# model training
bi_lstm_history = bi_lstm.fit(
    X_train, y_train,
    batch_size=BATCH_SIZE,
    validation_data = (X_test, y_test),
    callbacks = [EarlyStopping(monitor='val_loss',
    ↪  patience=PATIENCE)],
    epochs=NUM_EPOCHS)
```



# Outline

Introduction

Data & Methods
- Dataset
- Model Type & Architecture
- Word Embedding Models

Implementation
- Data Preprocessing
- Models Development

Evaluation
- Evaluation Metrics
- Evaluation Results

Conclusion

# Evaluation Metrics

- **Accuracy**: what fraction of items is classified correctly?
- $F_1$ **score**:
  - Incorporates precision and recall:
    - Precision: how many selected items are relevant?
    - Recall: how many relevant items are selected?
  - A variant of F-score with precision and recall equally important

# Models Performance

| Model | Embedding | Accuracy | $F_1(B)$ | $F_1(PB+B)$ |
|---|---|---|---|---|
| Vanilla LSTM | Google News | 0.42 | 0.38 | 0.84 |
| | GloVe Twitter | 0.41 | 0.31 | 0.78 |
| | GloVe Common Crawl | 0.44 | **0.46** | **0.93** |
| Stacked LSTM | Google News | 0.39 | 0.30 | 0.77 |
| | GloVe Twitter | 0.42 | 0.41 | 0.82 |
| | GloVe Common Crawl | 0.42 | 0.45 | **0.93** |
| Bi-LSTM | Google News | 0.42 | 0.33 | 0.75 |
| | GloVe Twitter | 0.44 | 0.34 | 0.83 |
| | GloVe Common Crawl | **0.46** | 0.37 | 0.85 |

Table: Models performance results. The best performance is indicated in bold

# Comparison with the Baseline

| Model type | Description | Accuracy | $F_1(B)$ | $F_1(PB+B)$ |
|---|---|---|---|---|
| Accuracy baseline | LSTM + word2vec Google News | 0.44 | 0.29 | 0.74 |
| Best accuracy score model | Bi-LSTM + GloVe Common Crawl | **0.46** | 0.37 | 0.85 |
| $F_1$ baseline | MemNN + attention | 0.29 | 0.36 | 0.87 |
| Best $F_1$ score model | Vanilla LSTM + GloVe Common Crawl | 0.44 | **0.46** | **0.93** |

Table: Comparison of the created models with the baseline models. The best performance is indicated in bold

# Relationship Exploration

- LSTM architecture in terms of model performance:
  1. Vanilla LSTM
  2. Stacked LSTM
  3. Bidirectional LSTM
- Word embedding model in terms of detector performance:
  1. GloVe Common Crawl
  2. GloVe Twitter
  3. word2vec Google News

# Patterns in Error Types

As the number of unknown tokens increase, the model performance decrease.

- Emoticons: simple (🙂 🙁) vs. complex (😎 😑)
- Apostrophe contractions:
  - standard, e.g., *isn′t* $\rightarrow$ *is not*
  - ambiguous, e.g., *he's* $\rightarrow$ *he has* or *he is*
- Misspellings:
  - standard, e.g., 'seee', 'tommorow'
  - merging words, e.g., 'questionWho', 'wait.Where'
- Abbreviations, e.g., *ConvAI* (Conversational Artificial Intelligence)
- Extra:
  - meaningless sequences of characters (e.g., *MAMAXMAMAX*)
  - tokens representing words from a different language

# Outline


Introduction

Data & Methods
- Dataset
- Model Type & Architecture
- Word Embedding Models

Implementation
- Data Preprocessing
- Models Development

Evaluation
- Evaluation Metrics
- Evaluation Results

Conclusion




KU LEUVEN

# Summary

First part:

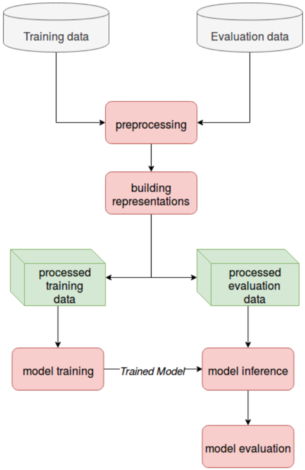
- dataset exploration
- existing models overview
- selection of:
  - baseline
  - model for experiments

Second part:

- dataset preprocessing
- model development

Third part:

- models evaluation
- comparison with the baseline
- exploration of error patterns



# Future Work

- Model architecture: investigate further architecture types, e.g., LSTM with attention
- Word embedding model: experiment with Embeddings from Language Model (ELMo)
- Dataset expansion: including different languages and modalities
- Working with the unknown tokens
- Submitting the model for dialogue breakdown detection challenge 4

Thank you for your attention!

# Bibliography I

Dialogue breakdown detection challenge 4. URL: https://sites.google.com/site/dialoguebreakdowndetection4/, last checked on 30-01-2019.

dbdc33. Dataset. URL: https://dbd-challenge.github.io/dbdc3/datasets, last checked on 24-01-2019.

Ryuichiro Higashinaka, Kotaro Funakoshi, Michimasa Inaba, Yuki Arase, and Yuiko Tsunomori. The dialogue breakdown detection challenge 2. *Proceedings of SIG-SLUD*, 2016.

# Bibliography II

Ryuichiro Higashinaka, Kotaro Funakoshi, Michimasa Inaba, Yuiko Tsunomori, Tetsuro Takahashi, and Nobuhiro Kaji. Overview of dialogue breakdown detection challenge 3. *Proceedings of Dialog System Technology Challenge*, 6, 2017.

Michimasa Inaba and Kenichi Takahashi. Improving the performance of chat-oriented dialogue systems via dialogue breakdown detection.

José Lopes. How generic can dialogue breakdown detection be? the kth entry to dbdc3.

Bilyana Martinovsky and David Traum. The error is the clue: Breakdown in human-machine interaction. Technical report, Institute for Creative Technologies, University of Southern California, 2006.

# Bibliography III

Higashinaka Ryuichiro, Kotaro Funakoshi, Yuka Kobayashi, and Inaba Michimasa. The dialogue breakdown detection challenge: Task description, datasets, and evaluation metrics. 2015.

Hiroaki Sugiyama. Utterance selection based on sentence similarities and dialogue breakdown detection on ntcir-12 stc task. In *NTCIR*, 2016.



KU LEUVEN

# Appendix

# Task Setting

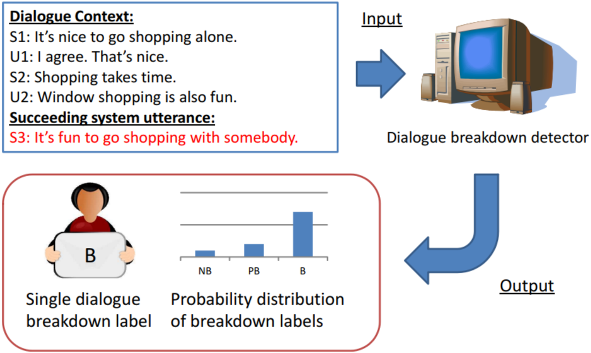



Figure: Task setting [Higashinaka et al., 2017]

# Dialogue Representation in JSON

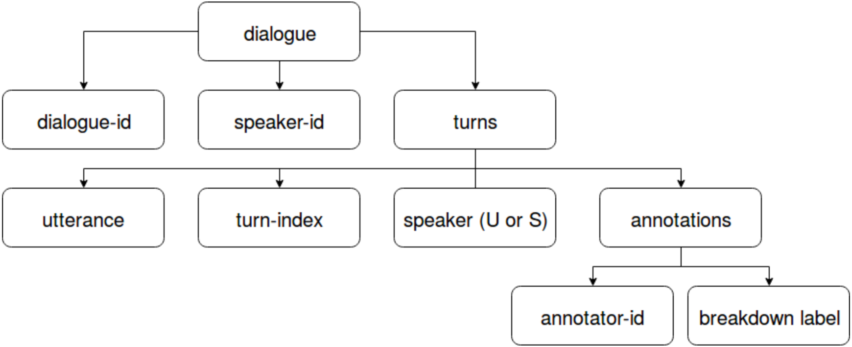



Figure: Dialogue representation in JSON: fields hierarchy


KU LEUVEN

# The Dataset Statistics

| | Development data | | | | Evaluation data | | | |
|---|---|---|---|---|---|---|---|---|
| | TKTK | IRIS | CIC | YI | TKTK | IRIS | CIC | YI |
| Dialogues | 100 | 100 | 115 | 100 | 50 | 50 | 50 | 50 |
| Annotators | CF | CF | AMT | AMT | CF | CF | AMT | AMT |
| NB | 35.1% | 32.9% | 28.9% | 34.8% | 44.3% | 34.5% | 29.1% | 35.4% |
| PB | 27.6% | 27.8% | 29.8% | 36.1% | 29.2% | 29.3% | 39.3% | 40.3% |
| B | 37.3% | 39.4% | 41.3% | 29.1% | 26.5% | 36.2% | 31.6% | 24.3% |
| Fleiss' $\kappa$ | 0.14 | 0.11 | 0.05 | 0.01 | 0.13 | 0.09 | 0.001 | -0.006 |

Table: The dataset statistics

# Types of Models

- Conditional Random Fields (CRF)
- Extremely Randomized Trees (ETR)
- Maximum Entropy model (MaxEnt)
- Support Vector Machines (SVM)
- Memory Networks (MemNN)
- Recurrent Neural Networks (RNN). Includes Long Short-term Memory Network (LSTM)
- Models with attention

| Model type | Implementation description | Accuracy | $F_1(B)$ | $F_1(PB+B)$ |
|---|---|---|---|---|
| CRF | baseline model for DBDC3 | 0.420 | 0.354 | 0.762 |
| MaxEnt | *MaxEnt, labelling exchanges | 0.410 | 0.240 | 0.220 |
| SVM | SVM + SpeDial feature set | 0.340 | 0.350 | 0.840 |
| MemNN | MemNN + attention | 0.295 | **0.364** | **0.874** |
| | MemNN + attention; trained on multiligual data | 0.290 | 0.356 | 0.870 |
| ETR | ETR + geometric mean | 0.426 | 0.312 | 0.832 |
| | ETR + cosine similarity of all word pairs | 0.431 | 0.320 | 0.840 |
| | ETR + arithmetic mean | 0.420 | 0.302 | 0.835 |

| Model type | Implementation description | Accuracy | $F_1(B)$ | $F_1(PB+B)$ |
|---|---|---|---|---|
| RNN | RNN + attention between sentences, GloVe Twitter | 0.360 | 0.208 | 0.346 |
| | RNN + attention + GloVe Twitter + finetuning | 0.210 | 0.210 | 0.340 |
| | RNN + attention + GloVe Twitter + extra linguistic features | 0.356 | 0.320 | 0.805 |
| | LSTM + BoW, word embeddings | **0.440** | 0.290 | 0.744 |
| | LSTM + BoW, document embeddings | 0.422 | 0.340 | 0.759 |
| | Hie-Bi-LSTM + GloVe Wikipedia | 0.429 | 0.321 | 0.763 |

Table: Models overview: classification results. The best performance is indicated in bold. An asterisk signifies submission without technical paper

# Datasets Preprocessing

Decisions to make before preprocessing:

- Keep user utterances? Yes, it improves model performance [Lopes]
- ow to feed user utterances? As separate utterances marked with $U$ label

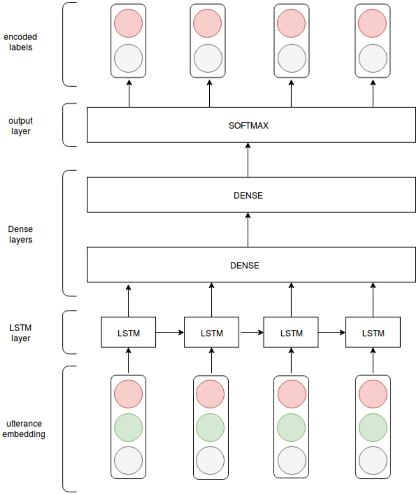
# Vanilla LSTM

```
# imports
from keras.models import Sequential
from keras.layers import LSTM, Dense

# model definition
vanilla = Sequential()
# input layer
lstm1 = LSTM(NUM_CELLS, input_shape=(MAX_UTTERANCE_LENGTH,
↪  UTTERANCE_DIMENSIONALITY), return_sequences=True, dropout =
↪  DROPOUT_COEFF, recurrent_dropout = DROPOUT_COEFF)
vanilla.add(lstm1)
# hidden layers
vanilla.add(Dense(NUM_CELLS, activation = 'relu'))
vanilla.add(Dense(NUM_CELLS, activation = 'relu'))
# output layer
outputs = Dense(NB_LABELS, activation = 'softmax')
vanilla.add(outputs)

# model compilation
vanilla.compile(
    optimizer = 'rmsprop',
    loss = 'categorical_crossentropy',
    metrics=['accuracy'])

# model training
vanilla_history = vanilla.fit(
    X_train, y_train,
    batch_size=BATCH_SIZE,
    validation_data = (X_test, y_test),
    callbacks = [EarlyStopping(monitor='val_loss', patience=5)],
    epochs=NUM_EPOCHS)
```



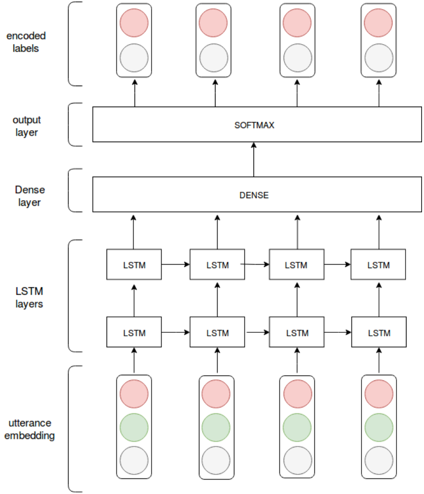
# Stacked LSTM

```
# imports
from keras.models import Sequential
from keras.layers import LSTM, Dense

# model definition
stacked = Sequential()
# input layer
lstm1 = LSTM(NUM_CELLS, input_shape=(MAX_UTTERANCE_LENGTH,
↪  UTTERANCE_DIMENSIONALITY), return_sequences=True, dropout =
↪  DROPOUT_COEFF, recurrent_dropout = DROPOUT_COEFF)
stacked.add(lstm1)
# hidden layers
lstm2 = LSTM(NUM_CELLS, input_shape=(MAX_UTTERANCE_LENGTH,
↪  UTTERANCE_DIMENSIONALITY), return_sequences=True, dropout =
↪  DROPOUT_COEFF, recurrent_dropout = DROPOUT_COEFF)
stacked.add(lstm2)
vanilla.add(Dense(NUM_CELLS, activation = 'relu'))
# output layer
stacked = Dense(NB_LABELS, activation = 'softmax')
stacked.add(outputs)

# model compilation
stacked.compile(
    optimizer = 'rmsprop',
    loss = 'categorical_crossentropy',
    metrics=['accuracy'])

# model training
stacked_history = stacked.fit(
    X_train, y_train,
    batch_size=BATCH_SIZE,
    validation_data = (X_test, y_test),
    callbacks = [EarlyStopping(monitor='val_loss',
    ↪  patience=PATIENCE)],
    epochs=NUM_EPOCHS)
```



# Relationship Exploration

| LSTM type | Accuracy | $F_1(B)$ | $F_1(PB+B)$ |
|---|---|---|---|
| Vanilla LSTM | 0.42 | 0.38 | 0.85 |
| Stacked LSTM | 0.41 | 0.39 | 0.84 |
| Bi-LSTM | 0.44 | 0.35 | 0.81 |

Table: Average metric scores for every LSTM type

| Embedding type | Accuracy | $F_1(B)$ | $F_1(PB+B)$ |
|---|---|---|---|
| word2vec Google News | 0.41 | 0.34 | 0.78 |
| GloVe Twitter | 0.42 | 0.35 | 0.81 |
| GloVe Common Crawl | 0.44 | 0.43 | 0.90 |

Table: Average metric scores for every embedding type

# Answers to Research Questions

- There are several major types of dialogue breakdown detectors: CRF, ETR, MaxEnt, SVM, MemNN, RNN, and models with attention
- Comparison of the existing models allow concluding that LSTM and MemNN appear to produce better results
- Both model architecture and word embedding model influence the performance. Word embedding model appear to produce a more significant impact.