CODETE

Building intelligent bots

Implementing rule-, retrieval-, and generative-based bots using NLP tools

---

Karol Przystalski

09.07.2018

About me

Overview

2015 – obtained a Ph.D. in Computer Science @ Polish Science Academy and Jagiellonian University

2010 until now – CTO @ Codete

2007 - 2009 – Software Engineer @ IBM

Recent research papers

Multispectral skin patterns analysis using fractal methods, K. Przystalski and M. J.Ogorzalek. Expert Systems with Applications, 2017

`https://www.sciencedirect.com/science/article/pii/S0957417417304803`

Contact

karol@codete.comhttps://

0048 608508372

Environment

https://hub.docker.com/r/kprzystalski/codete_ml_workshops/

https://github.com/codete/oreilly

Agenda

1. Introduction
2. Rule-based chatbots
3. Retrieval-based
4. Generative-based
5. Summary

Introduction

Chatbots – a new interface

Bots are a new way of communication between the user and the app [1].

[1] Designing Bots, 1st Edition. *Amir Shevat*, O'Reilly Media 2017

Bot taxonomy

Bots can be divided into a few types, based on:

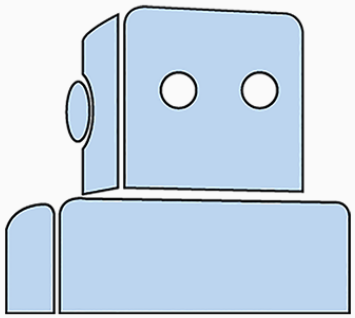
- interface – automation, audio or text,
- privacy – on-site and online,
- usage – superbots, domain-driven, etc.[2]

[1] Designing Bots, 1st Edition. *Amir Shevat*, O'Reilly Media 2017

Implementations

{LawGeex}

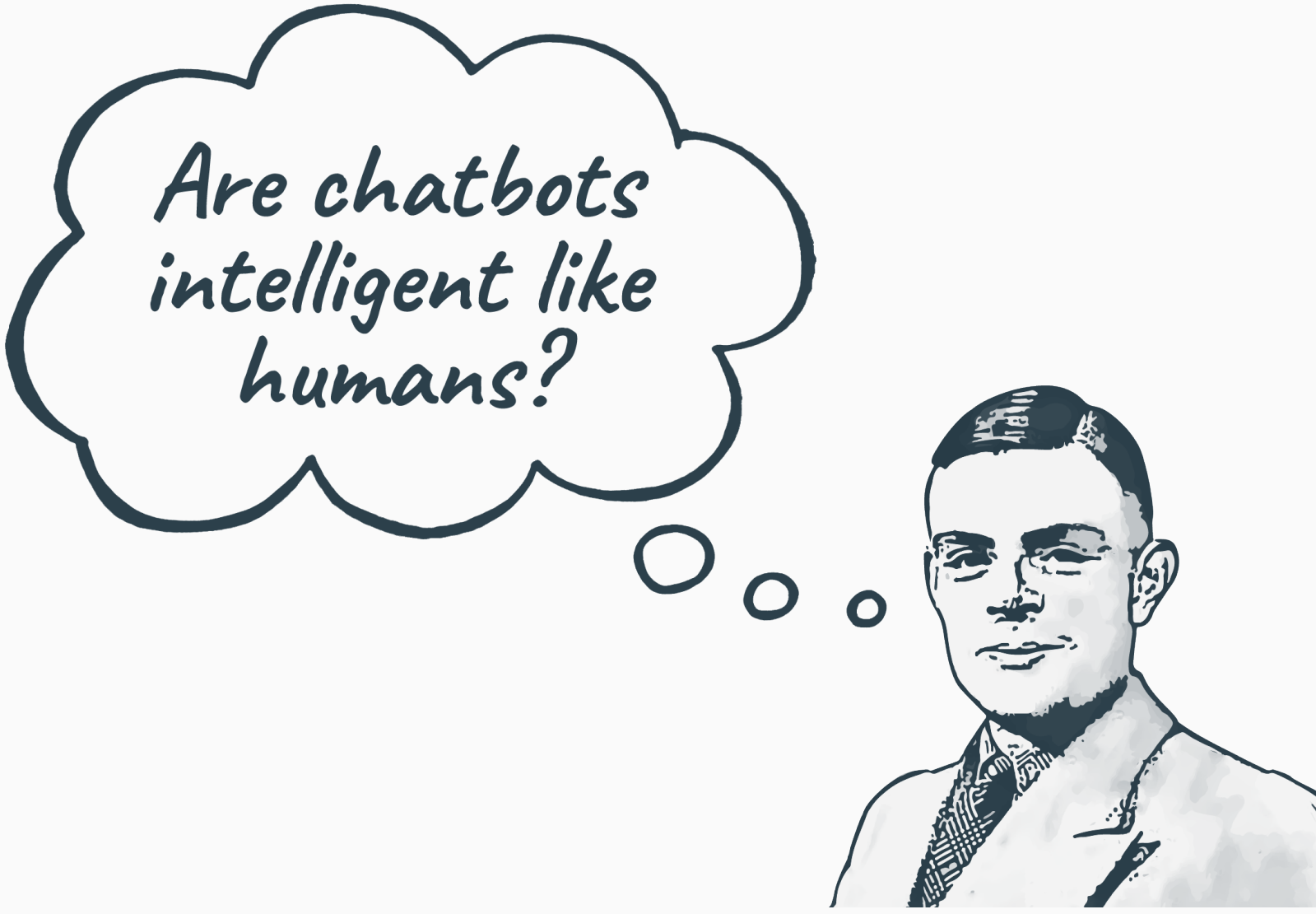
Tools



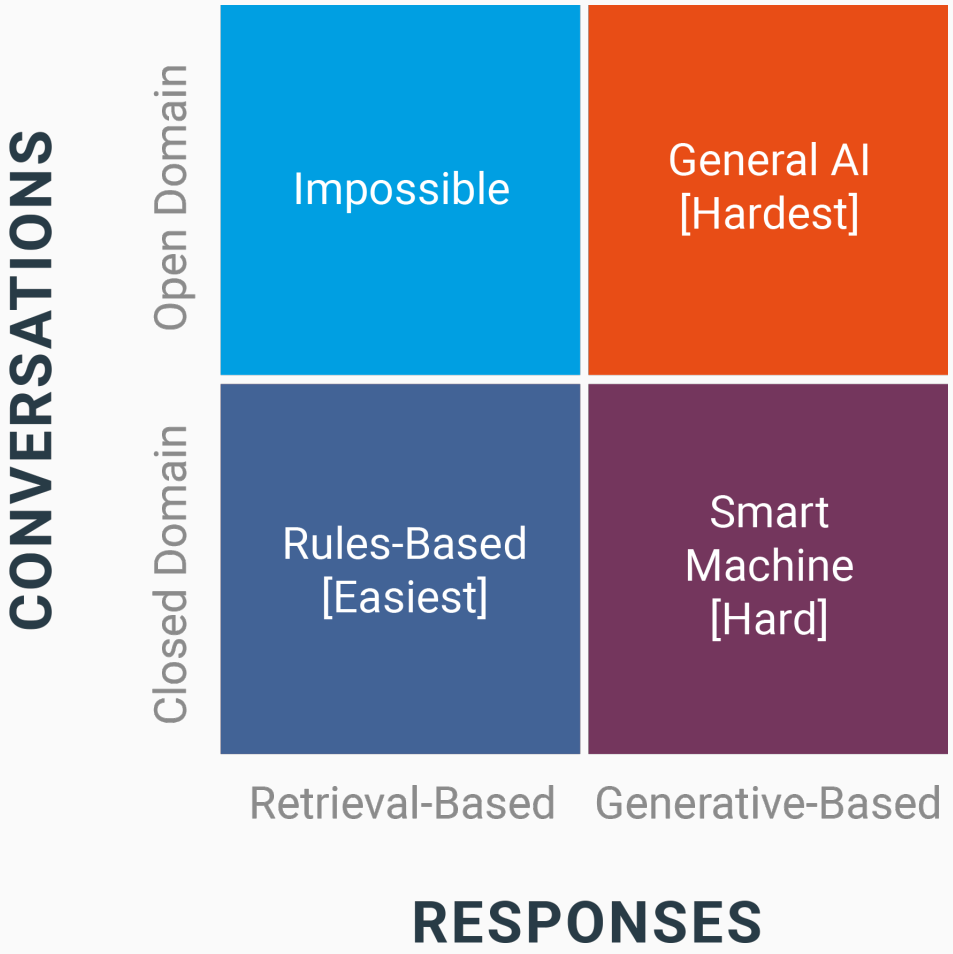
Integration

You can find a short explanation on how to start in the chatbots notebooks:

`https://github.com/codete/oreilly/blob/master/Chatbots.ipynb`


Are chatbots intelligent like humans?

Bot matrix



[1]Ultimate Guide to Leveraging NLP and Machine Learning for your Chatbot. Stefan Kojouharov, Chatbots Life 2016

Rule-based chatbots

Rule-based – basics

Phrases list	Answers list
Show status of recruitment.	We have currently X candidates.
What is the weather in Berlin?	It is <current weather>.
Hi!	Hi. How are you?
Hire candidate <name>.	Sent an email to <email>.

No valid phrase found

Regular expressions in Python for string comparison

Regular expressions in Python are almost the same as in any other programming languages. We can use regex methods to:

- search – finds only the first occurrence of expression in text,
- match – finds all occurrences of expression in text,
- fullmatch – matches only if the whole string matches the regular expression,
- split – splits into a list based on the splitting expression,
- escape – replaces all characters in the pattern.

Regular expressions are used within many methods that we go through in the next slides.

Word and sentence comparison methods

String comparison methods available in Python:

- Levenshtein distance,
- Damerau-Levenshtein distance,
- Jaro distance,
- Jaro-Winkler distance,
- Match rating approach comparison,
- Hamming distance,
- Gestalt pattern matching.

You can use at least two libraries:

- `Difflib` – `https://docs.python.org/3.6/library/difflib.html`,
- `Jellyfish` – `https://pypi.org/project/jellyfish/`.

String comparison – Levenshtein Distance

The Levenshtein distance is a number of insertion, deletion or replacement changes that needs to be done to get the same strings.

It is a number that is equal or higher than 0. It can be normalized to get a number from 0 to 1.

compared words	word length
train*ing*	8
train*s*	6

The distance for both words is 3. After the normalization the distance is $\frac{3}{8} = 0.375$.

String comparison – Gestalt pattern matching

This solution can be formulated as:

$$G_{PM} = \frac{\#\text{same characters}}{\#\text{total characters}}.$$

For the same example we have 5 same characters in each word and four that are different. This makes the G_{PM} value:

$$G_{PM} = \frac{10}{14} = 0.7142.$$

SQL Like vs. Full-text search

The full-text search is in most cases much faster than a Like query.

$$\text{bm25}(D, Q) = -1 \sum_{i=1}^{n} \text{IDF}(q_i) \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \frac{|D|}{\text{avgl}})},$$

where:

- $|D|$ is the number of tokens in the current document,
- k_1 and b are constants with values 1.2 and 0.75,
- avgl is the average number of tokens.

SQL Like vs. Full-text search

IDF is the inverse-document-frequency of query phrase i and is formulated as:

$$\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5},$$

where:

- N is the total number of rows in table,
- $n(q_i)$ is the total number of rows that contain at least one instance of phrase i.

$f(q_i, D)$ is the phrase frequency of phrase i:

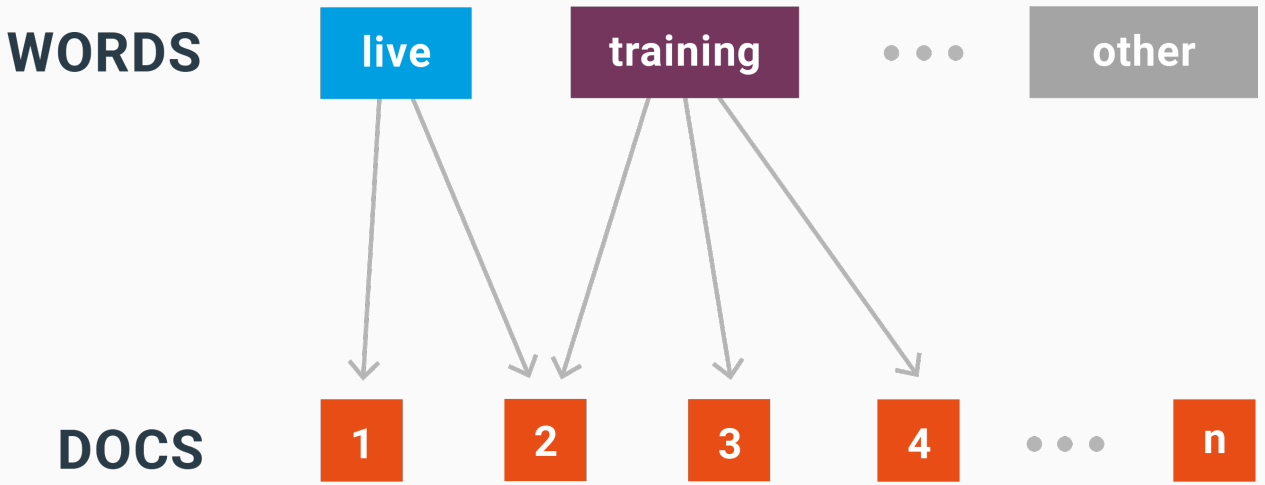
$$f(q_i, D) = \sum_{1}^{nc} w_c \cdot n(q_i, c),$$

where:

- w_c are the weights assigned to columns,
- $n(q_i, c)$is the number of occurrences of phrase i in column c of the current row.

SQL Full-text search

This is a **live** **training**.



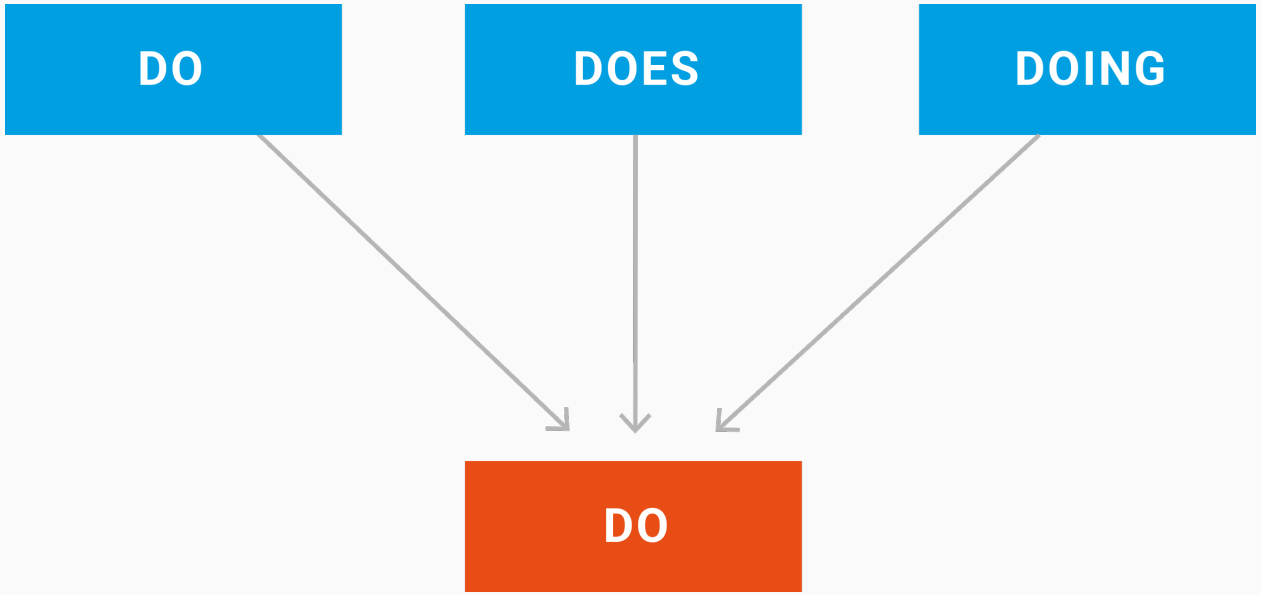
NLP methods used for sentence comparison

There are three popular methods that are used in rule-based chatbots:

- tokenization,
- lemmatization,
- stemming.

Tokenization divides a sentence into separate words.

Lemmatization and stemming



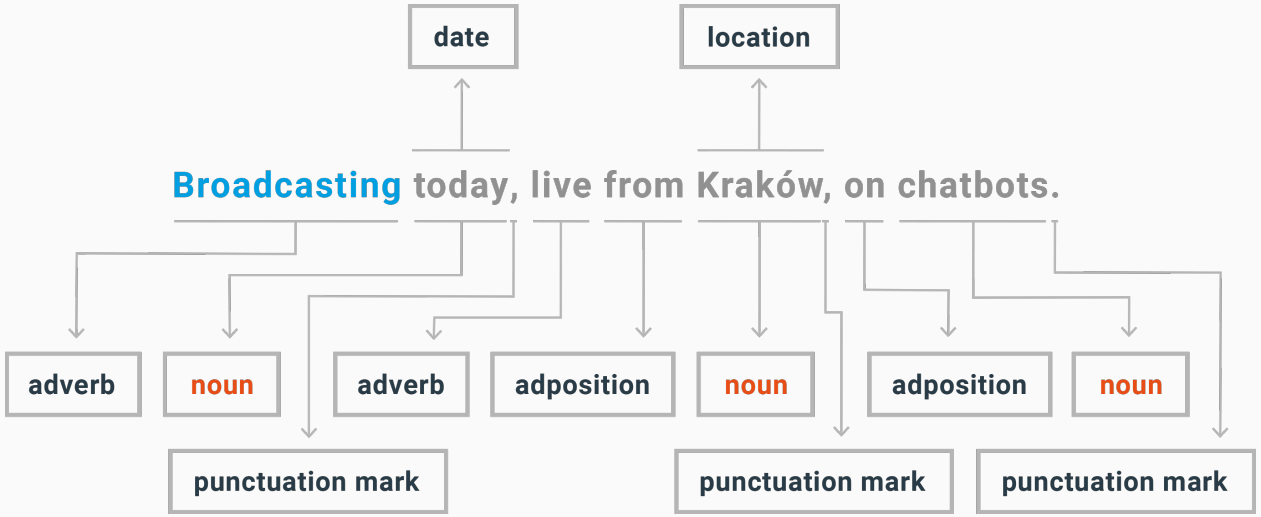
Retrieval-based

Natural Language Understanding

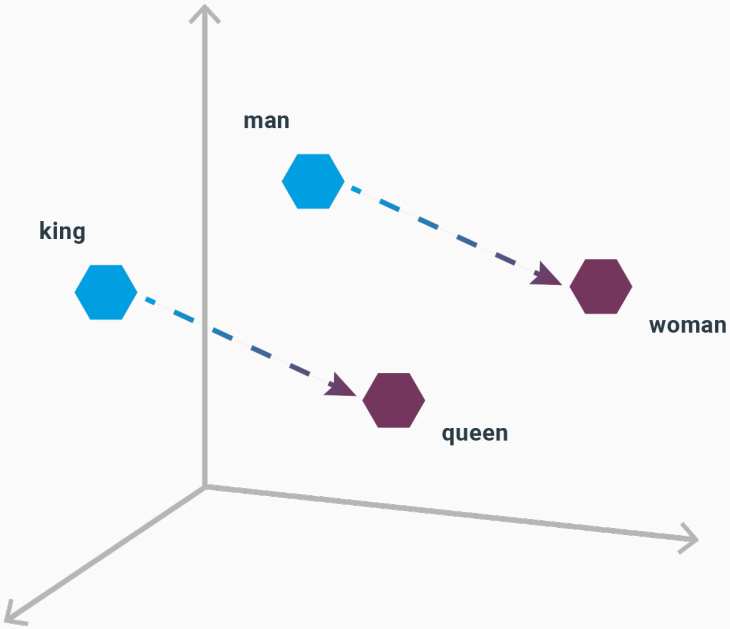
Natural Language Understanding is a part of Natural Language Processing. NLU uses NLP methods to understand what the text is about. There are three popular NLP methods that make it easier to understand written text:

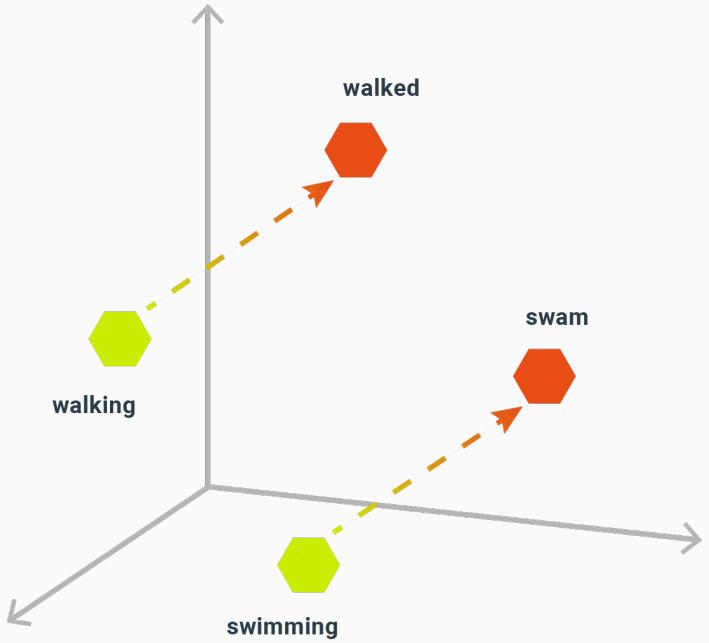
- part of speech,
- noun chunk,
- named entity recognition.

Retrieval-based – NLU

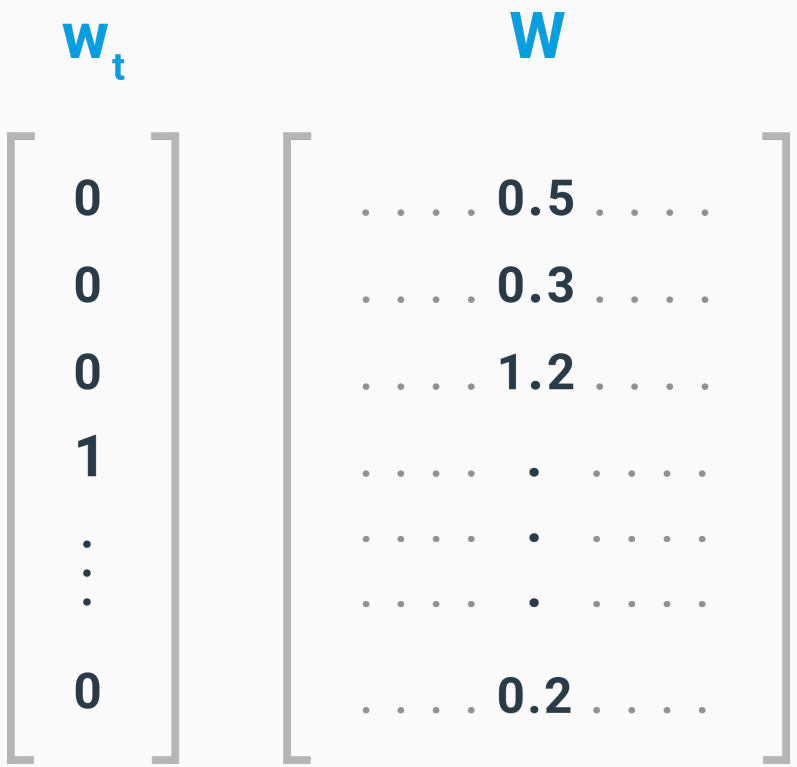


Word vectorization





Word vectorization – concept

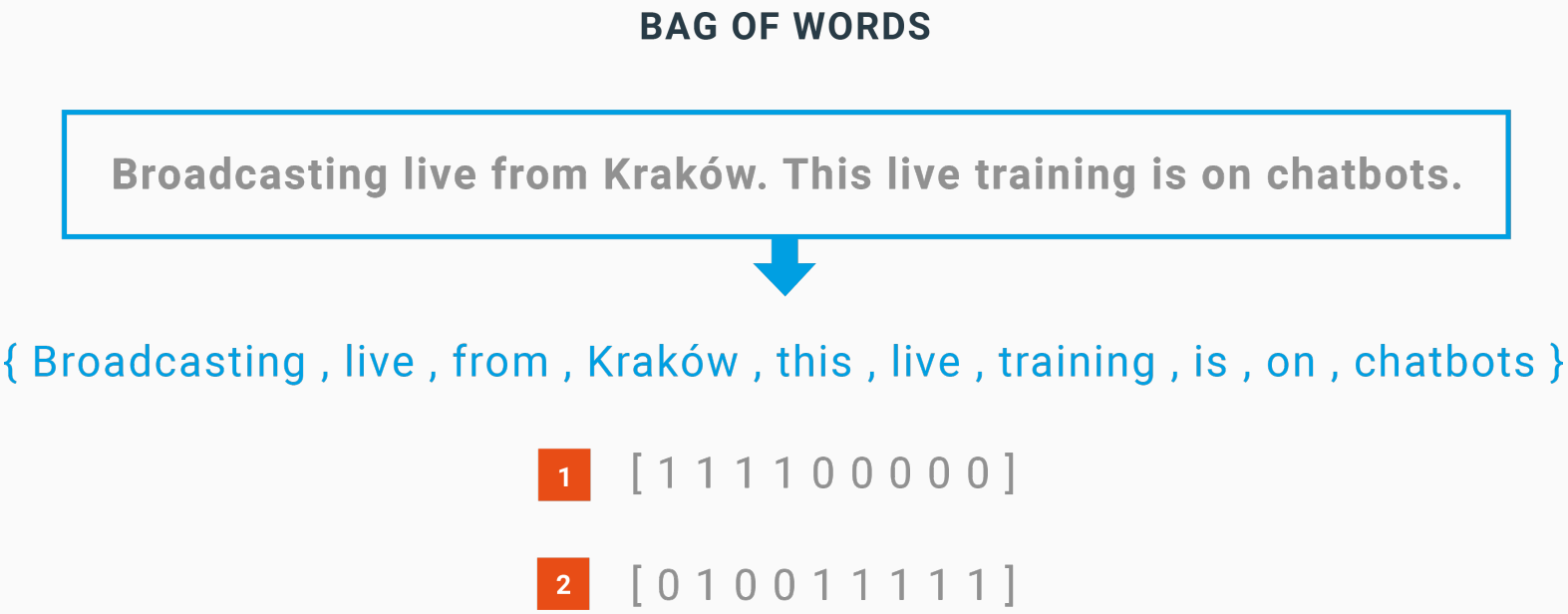


Word vectorization – methods

The most popular methods that are used to create a space of vectorized words are:

- bag of words,
- tf-idf,
- transfer learning,
- n-gram model,
- skip-thought vectors.

Bag of words


BAG OF WORDS
Broadcasting live from Kraków. This live training is on chatbots.
{ Broadcasting , live , from , Kraków , this , live , training , is , on , chatbots }
1
[1 1 1 1 0 0 0 0 0]
2
[0 1 0 0 1 1 1 1 1]

Distance metrics

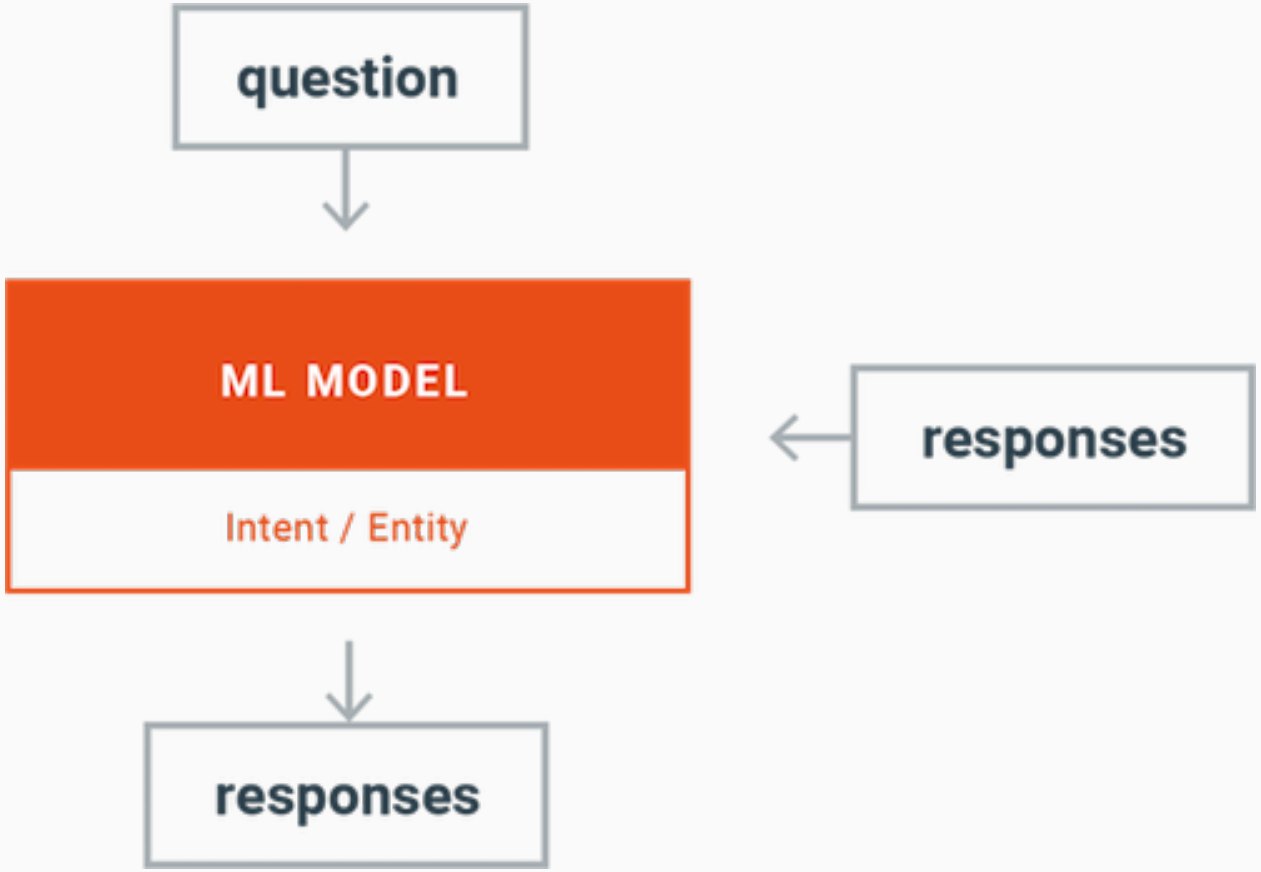
Also known as similarity or dissimilarity measures.

Measure name	equation	
Manhattan distance	$\rho_{Man}(x_r, x_s) = \sum_{i=1}^{n} \lvert x_{ri} - x_{si} \rvert$	(1)
Chebyshew distance	$\rho_{Ch}(x_r, x_s) = max_{1 \leq i \leq n} \lvert x_{ri} - x_{si} \rvert$	(2)
Frecht distance	$\rho(x_r, x_s) = \sum_{i=1}^{d} \frac{\lvert x_{ri} - x_{si} \rvert}{1 + \lvert x_{ri} + x_{si} \rvert} \frac{1}{2^i}$	(3)
Canberra distance	$\rho(x_r, x_s) = \sum_{i=1}^{d} \frac{\lvert x_{ri} - x_{si} \rvert}{\lvert x_{ri} + x_{si} \rvert}$	(4)
Post office distance	$\rho_{pos}(x_r, x_s) = \begin{cases} \rho_{Min}(x_r, 0) + \rho_{Min}(0, x_s), & \text{for } x_r \neq x_s, \\ 0, & \text{for } x_r = x_s \end{cases}$	(5)
Bray-Curtis distance	$\rho_{bc}(x_r, x_s) = \frac{\sum_{i=1}^{d} \lvert x_{ri} - x_{si} \rvert}{\sum_{i=1}^{d} (x_{ri} - x_{si})}$	(6)

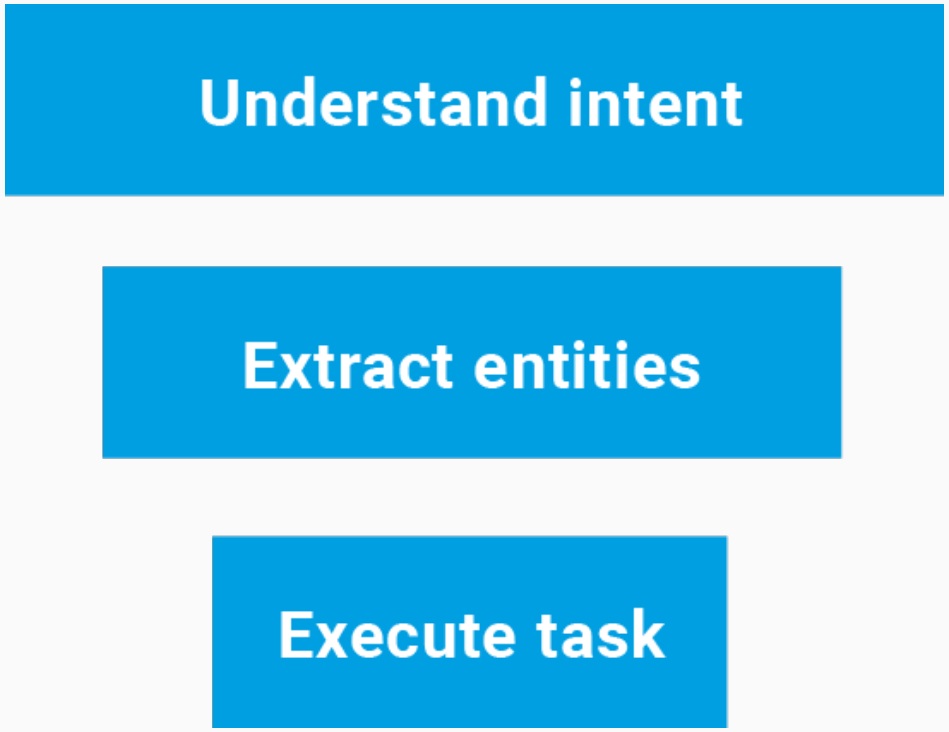
Tools

There are many tools that can be used to for NLU and retrieval-based chatbots.

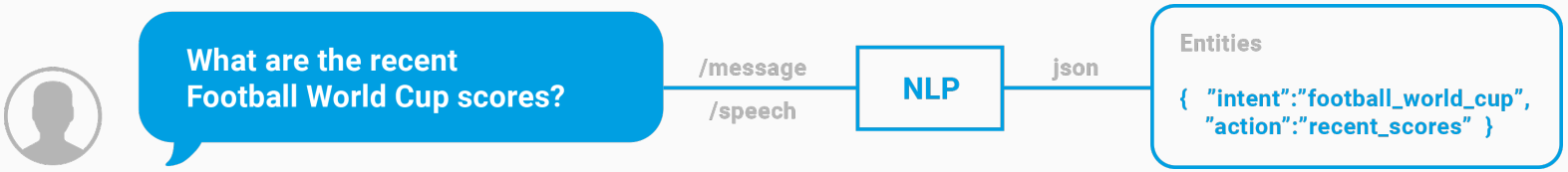
Retrieval-based – basics



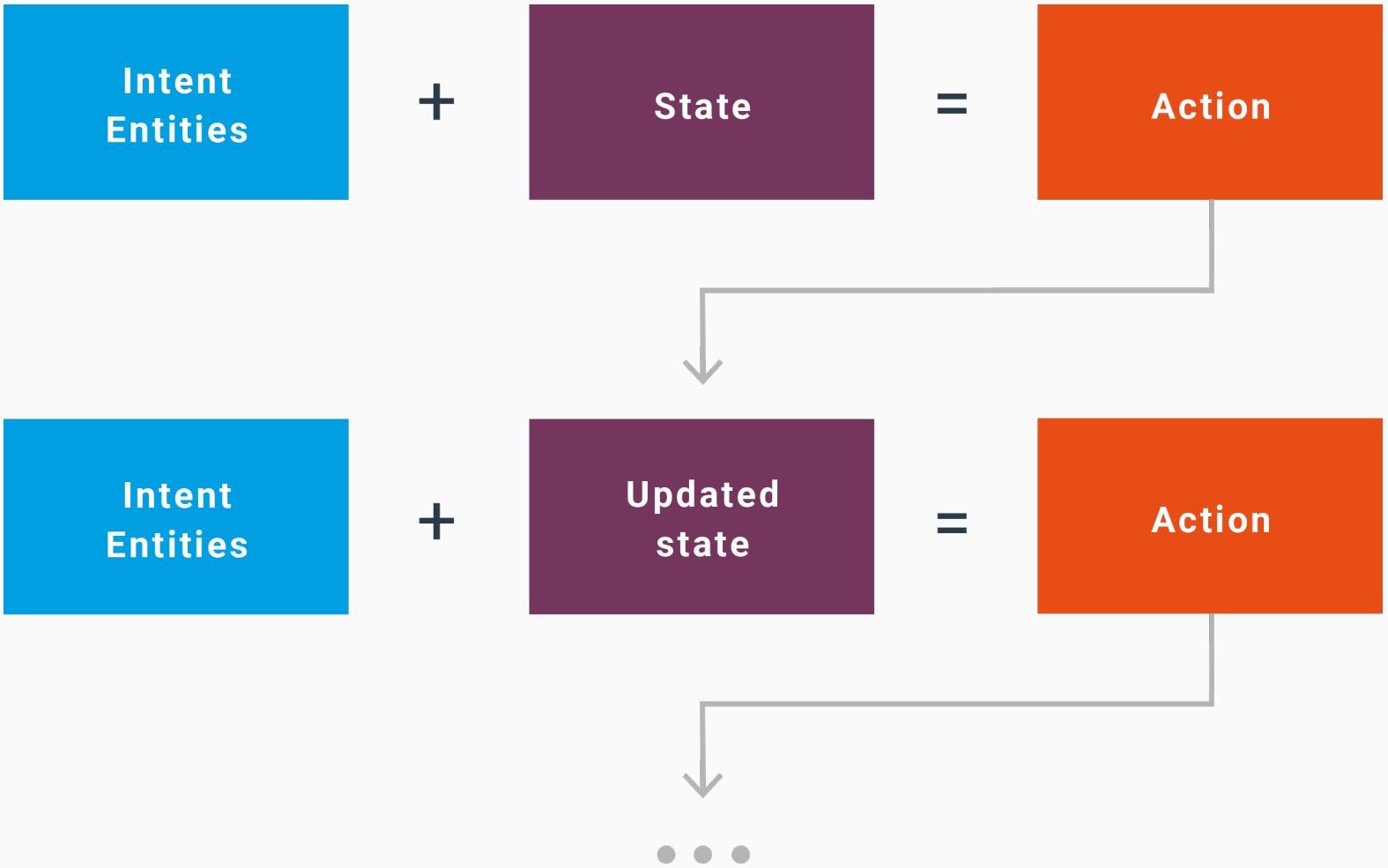
Entities and intents



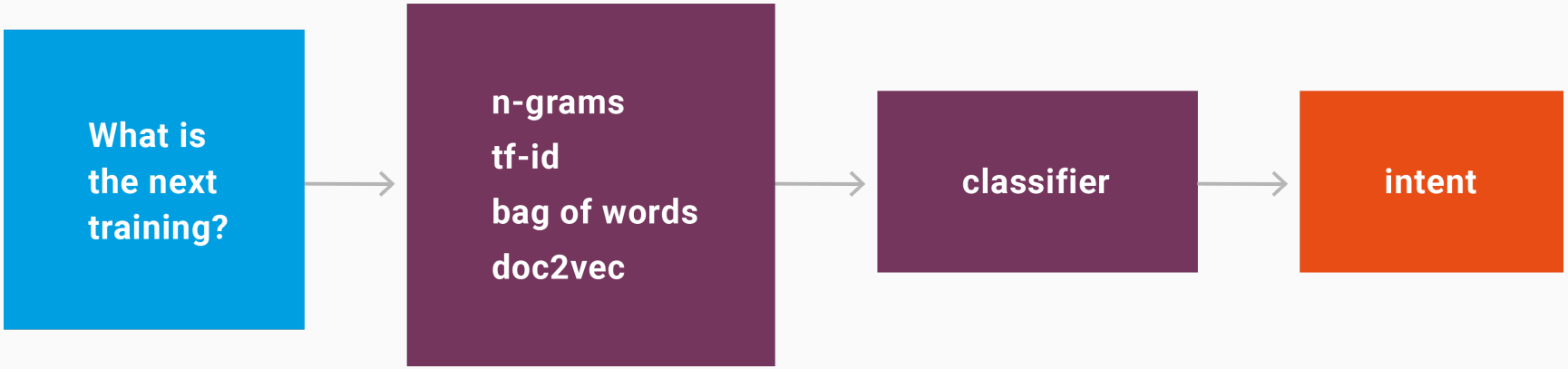
Entities and intents



Rasa NLU engine



Rasa intent learning process



Generative-based

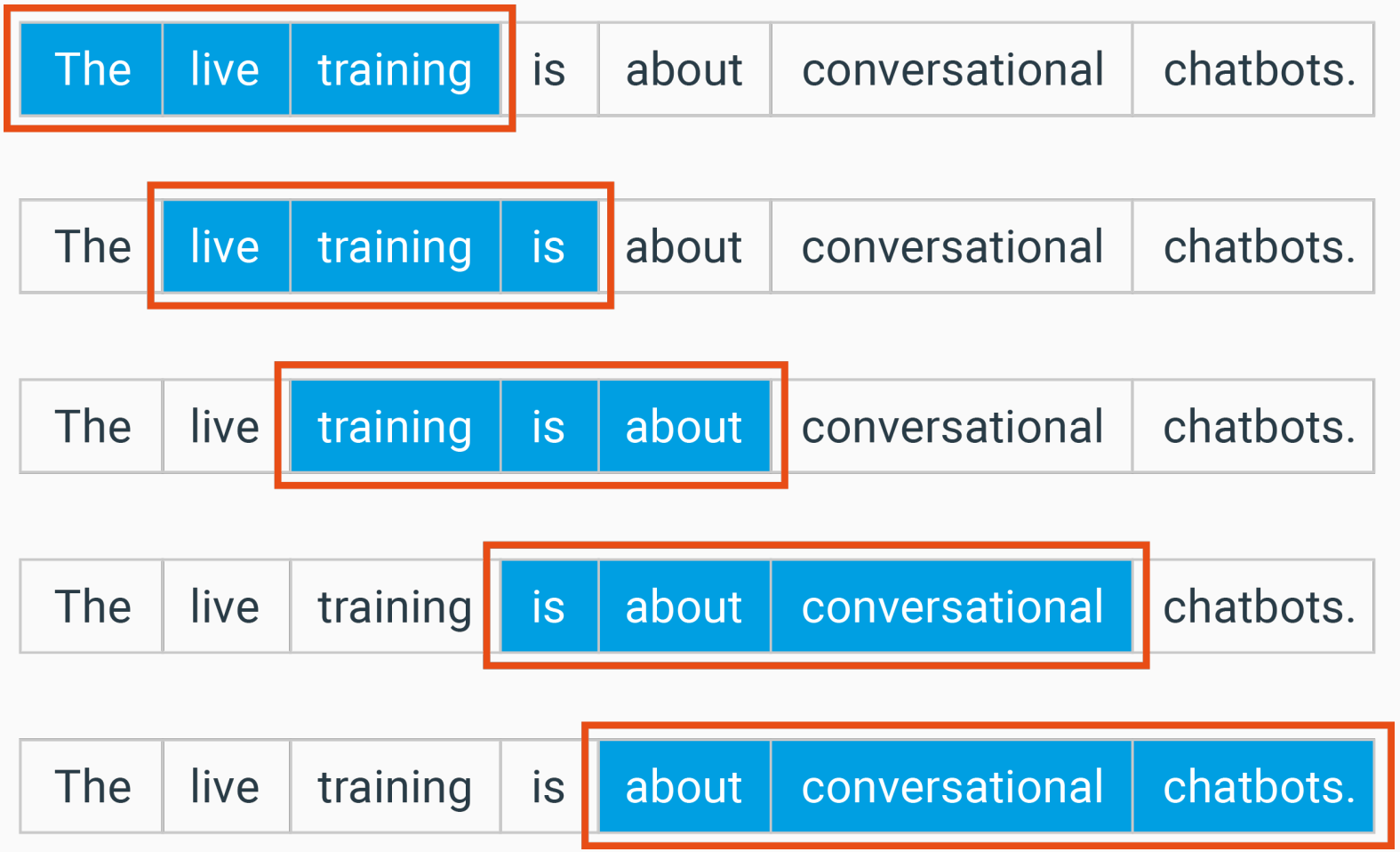
NLG

Natural Language Generation is a part of Natural Language Processing. The goal of NLG is to generate a sentence or the whole document that has a logical sense, follows the grammar and answers the question properly if we deal with a bot.

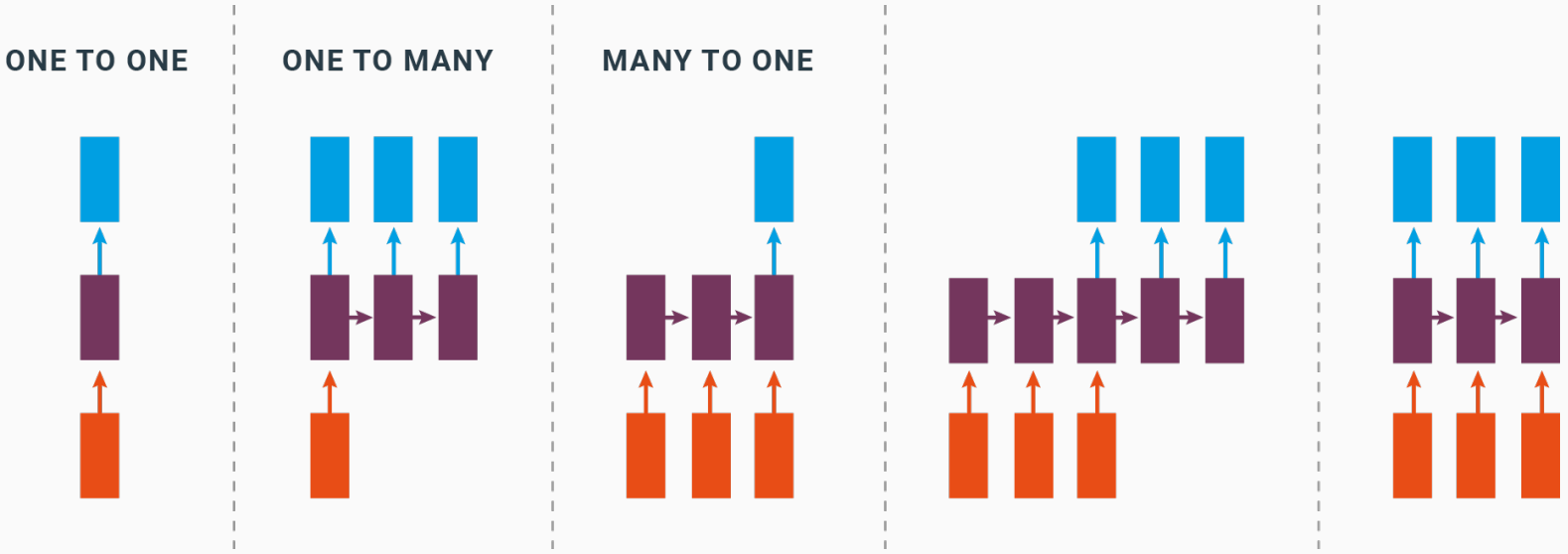
There are plenty of methods that can be used for text generation. The most popular are:

- n-gram model,
- recurrent neural network,
- autoencoders,
- generative adversarial network.

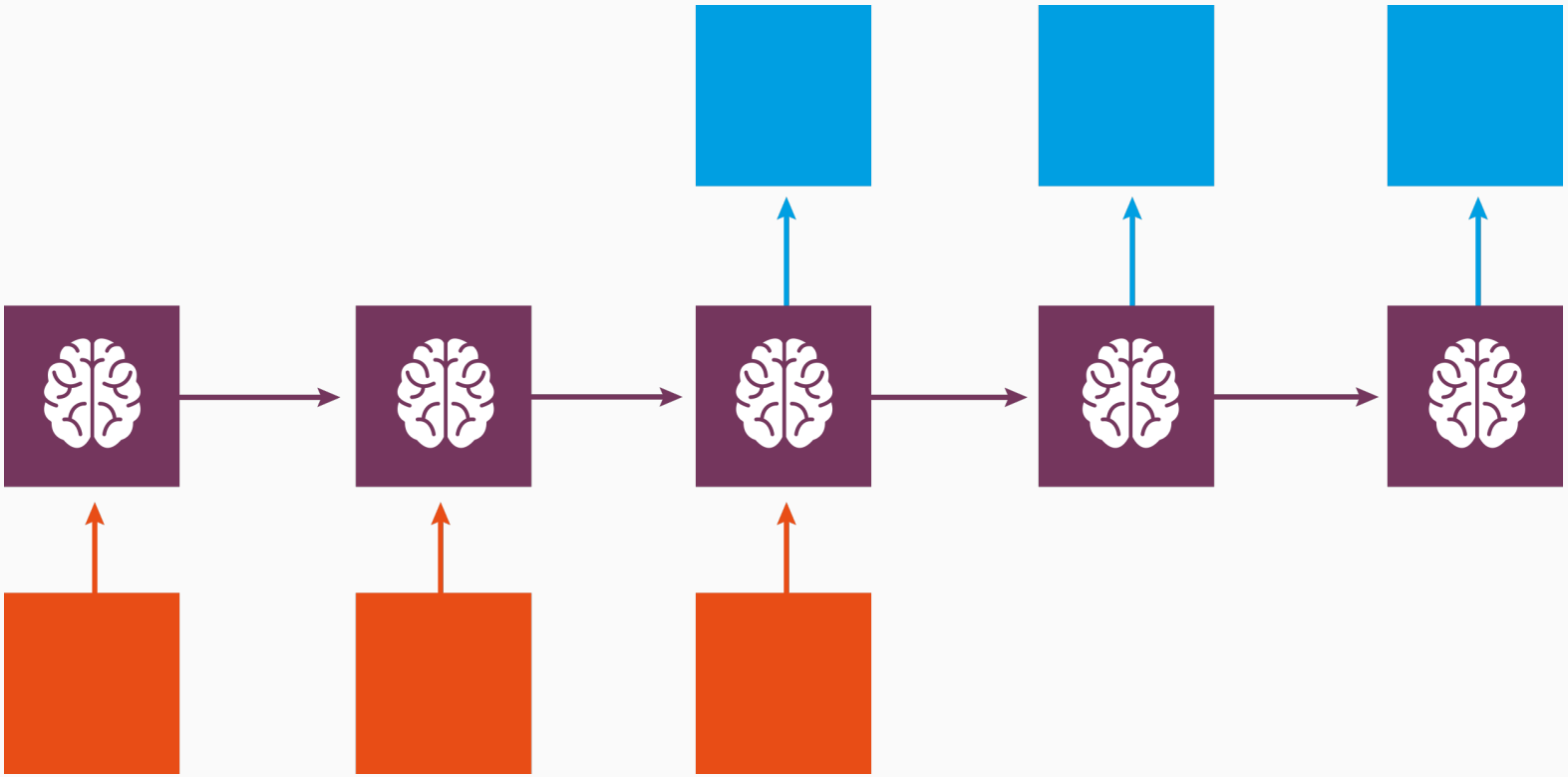
N-gram model


The live training is about conversational chatbots.
The live training is about conversational chatbots.
The live training is about conversational chatbots.
The live training is about conversational chatbots.
The live training is about conversational chatbots.

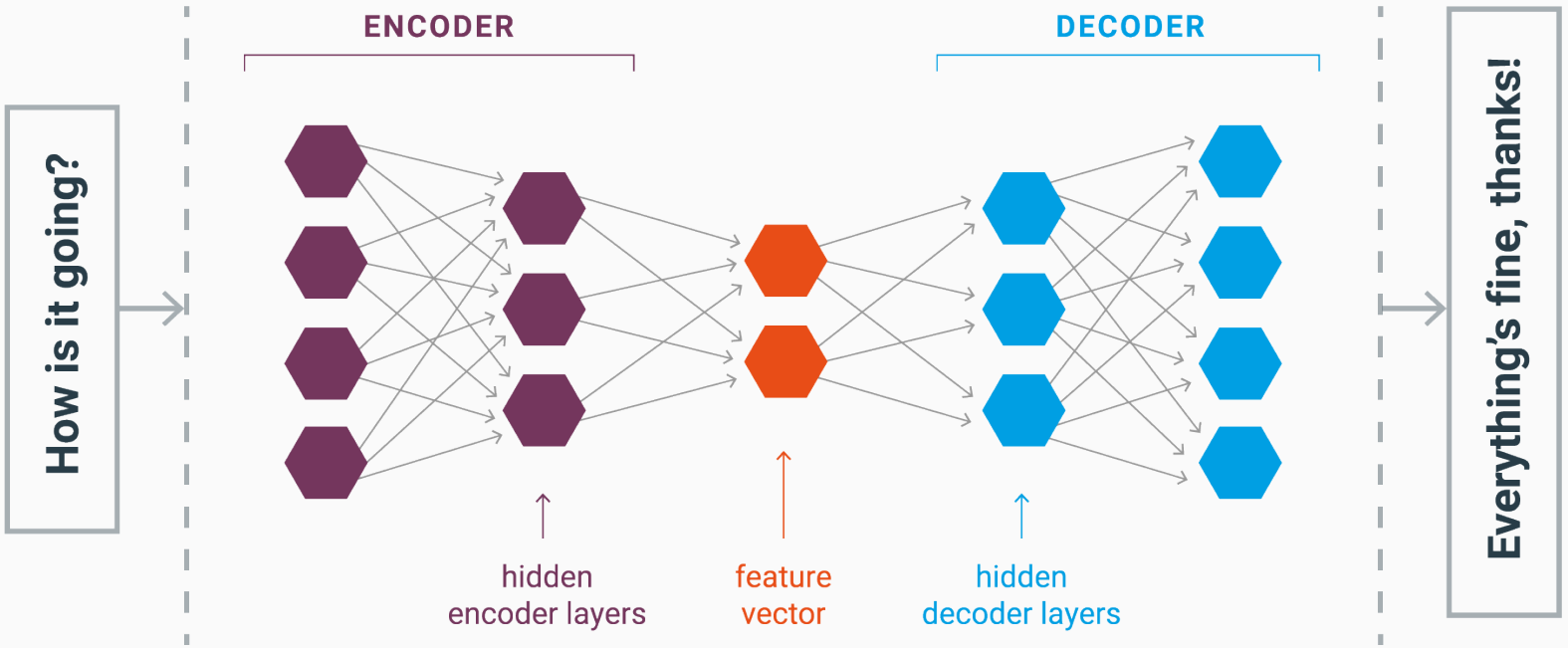
RNN



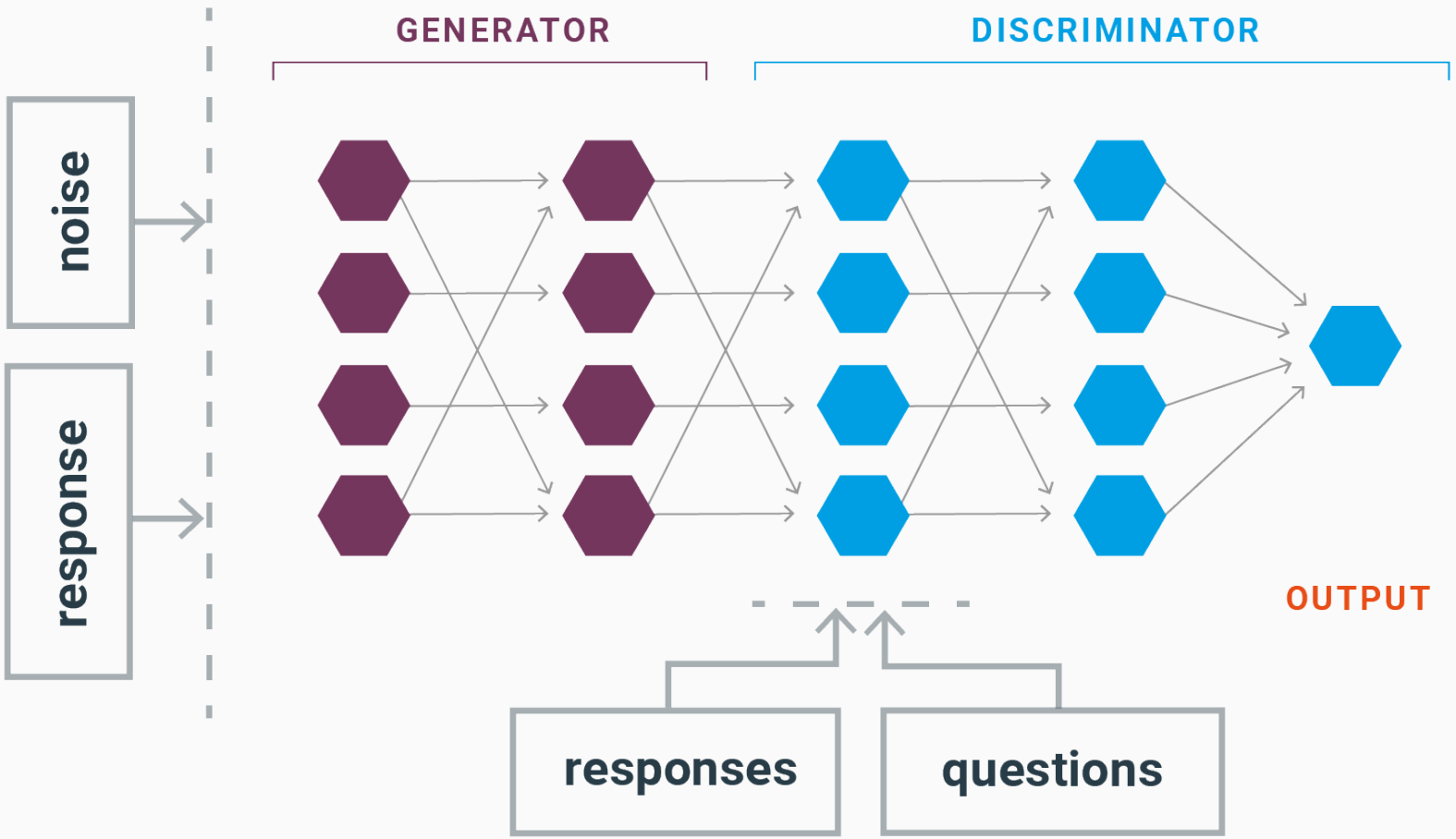
LSTM



Autoencoders



Generative Adversarial Networks



Working examples

There are many open source chatbots available. Here are a few worth mentioning:

- chatterbot – a chatbot implementation
  `http://chatterbot.readthedocs.io/`
- DeepQA – uses RNN and has a web interface
  `https://github.com/Conchylicultor/DeepQA`
- Generative Conversational Agents – uses LSTM, RNN and GAN
  `https://github.com/oswaldoludwig/Adversarial-Learning-for-Generative-Conversational-Agents`.

Research datasets

A few datasets useful for your research:

- **SQuAD** – reading comprehension dataset, consists of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to each question is a segment of text, `https://rajpurkar.github.io/SQuAD-explorer/`,
- **Cornell Movie Dialogs Corpus** – movie dialogs, `https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html`,
- **DeepMind datasets** – AQua is a dataset of questions and answers, `https://github.com/deepmind/AQuA`, more datasets from DeepMind: `https://deepmind.com/research/open-source/open-source-datasets/`,
- **DMQA** – Daily Mail and CNN articles data sets, `https://cs.nyu.edu/~kcho/DMQA/`,
- **MS MARCO** – Microsoft MAchine Reading COmprehension Dataset, `http://www.msmarco.org/dataset.aspx`.

Summary

Advantages

Rule-based chatbots:

- predictable,
- clear principles,
- cheap.

Retrieval-based chatbots:

- identify the intent,
- usually easy to train,
- do not need too many questions/answers,
- more intelligent than rule-based.

Generative-based chatbots:

- generic, intelligent answers,
- raw data as training data set.

Bottlenecks

Rule-based chatbots:

- too simple for most cases,
- not really intelligent.

Retrieval-based chatbots:

- limited to questions/answers
- not a generic solution.

Generative-based chatbots:

- usually take longer to train,
- needs a dataset, usually a huge one,
- sometimes unpredictable.

Where to go next?

Depending on your goal, we recommend to use one of presented architectures and use it with your dataset. Some hints on datasets:

- if you don't have any, you can generate some using two available chatbots like Alexa and the API to connect two chatbots together, let them speak and save the answers,
- double check your dataset and make sure you have cleaned it up,
- don't use the whole dataset in the first run of your solution, try it in smaller parts; especially when you use a deep architecture.

Feel free to join us at the presentation about sentiment analysis on July 11th.

Questions?

References i

William Fedus, Ian J. Goodfellow, and Andrew M. Dai.

Maskgan: Better text generation via filling in the _____.

CoRR, 2018.

M. Feng, B. Xiang, M. R. Glass, L. Wang, and B. Zhou.

Applying deep learning to answer selection: A study and an open task.

In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 813–820, 2015.

M. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger.

From word embeddings to document distances.

In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, pages 957–966, 2015.

References ii

Tan M, B. Xiang, and B. Zhou.

Lstm-based deep learning models for non-factoid answer selection.

CoRR, 2015.

J. Ratcliff and D. Metzener.

Pattern matching: The gestalt approach.

Dr. Dobb's Journal, page 46, 1999.

T-H. Wen, D. Vandyke, N. Mrkšić, M. Gasic, L. M. Rojas Barahona, P-H. Su, S. Ultes, and S. Young.

A network-based end-to-end trainable task-oriented dialogue system.

References iii

In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449. Association for Computational Linguistics, 2017.

H. Weng, Z. Qin, and T. Wan.

Text generation based on generative adversarial nets with latent variables, 2018.

W. Yin, H. Schutze, B. Xiang, and B. Zhou.

Abcnn: Attention-based convolutional neural network for modeling sentence pairs.

Transactions of the Association for Computational Linguistics, pages 259–272, 2016.

References iv

Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena.

Self-attention generative adversarial networks.

CoRR, 2018.

Y. Zhang, Z. Gan, and L. Carin.

Generating text via adversarial training.

Workshop on Adversarial Training, NIPS, 2016.