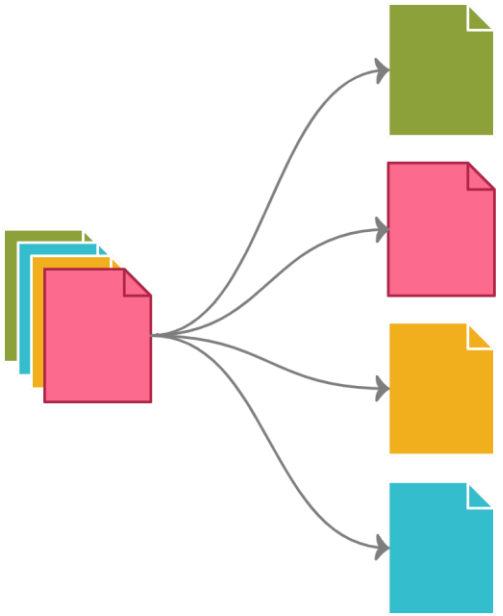
Text categorization, intent detection, and AI chatbots

Prof. Luca Cagliero
Dipartimento di Automatica e Informatica
Politecnico di Torino

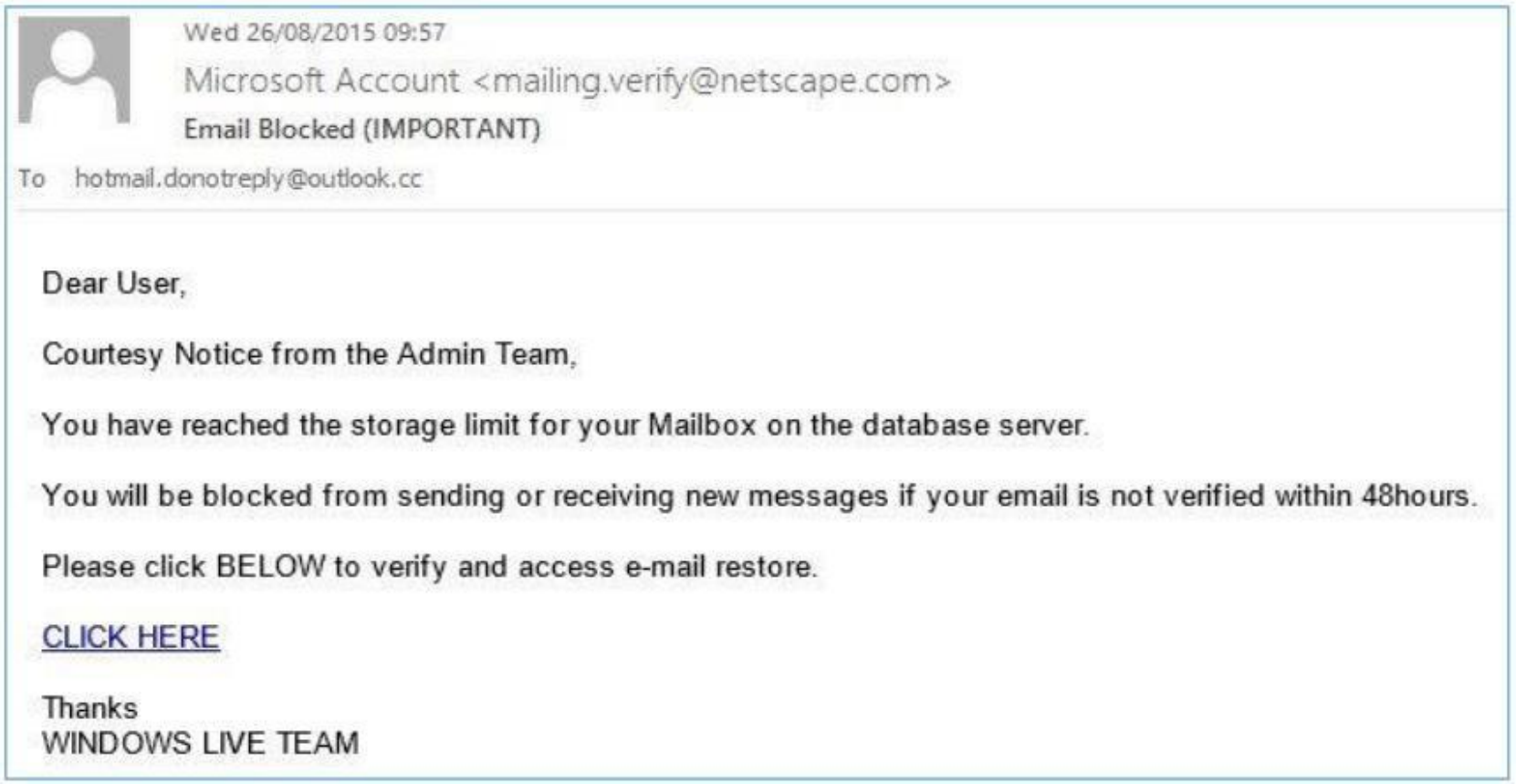
Politecnico di Torino

Text categorization

- Problem statement
  - It is the task of classifying unknown text into one or more predefined classes
    - Also denoted as text classification
- Supervised task
  - It requires the availability of a set of documents for which the classes are known
- Application scenarios
  - Spam detection
  - Topic classification
  - Sentiment analysis
  - Language identification



Applications: spam detection



Wed 26/08/2015 09:57

Microsoft Account <mailing.verify@netscape.com>

Email Blocked (IMPORTANT)

To hotmail.donotreply@outlook.cc

Dear User,

Courtesy Notice from the Admin Team,

You have reached the storage limit for your Mailbox on the database server.

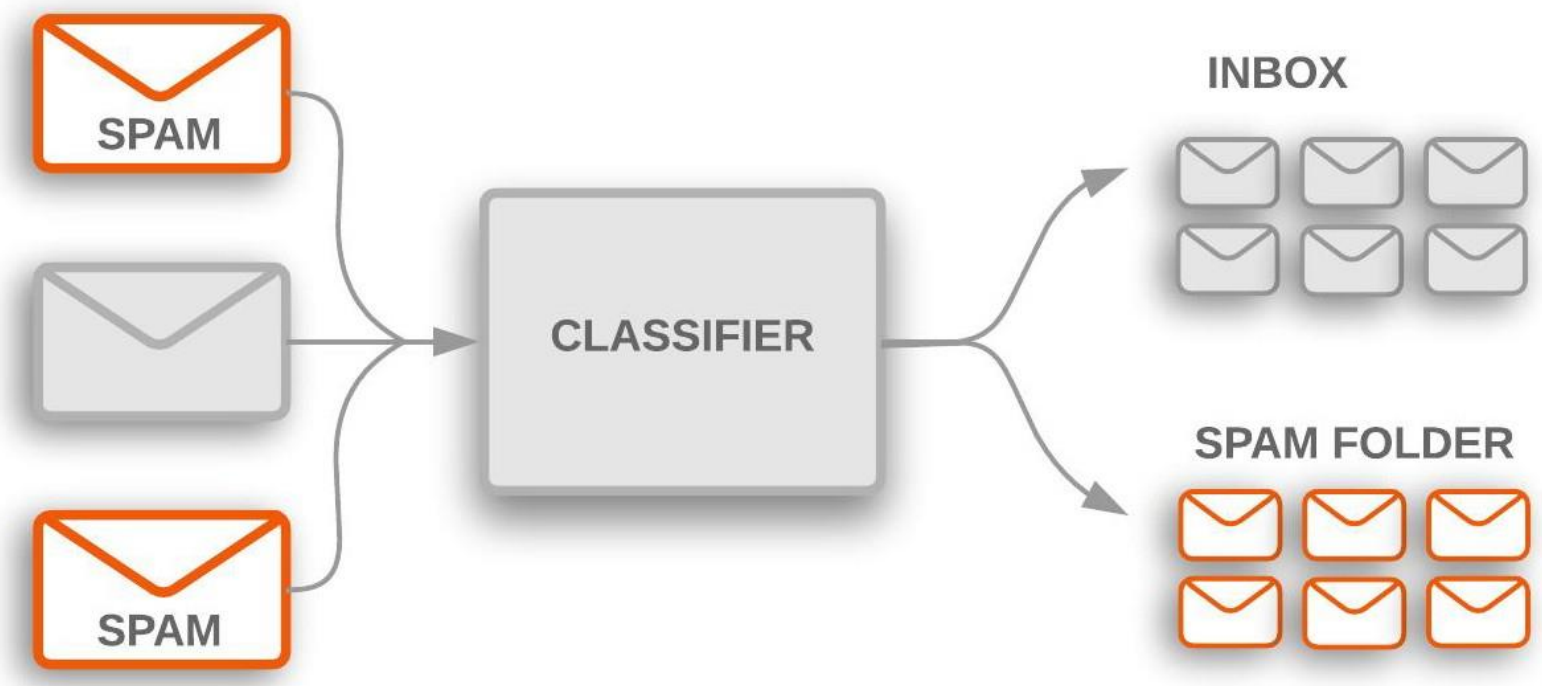
You will be blocked from sending or receiving new messages if your email is not verified within 48hours.

Please click BELOW to verify and access e-mail restore.

CLICK HERE

Thanks
WINDOWS LIVE TEAM

Applications: spam detection



Source: https://developers.google.com/ (latest access: April 2021)

Applications: sentiment analysis

- The movie was amazing! 
- It was pathetic, how can they ask to pay for that? 
- I lost 2 hours of my time 
- Well done! I can't wait for the second episode. 

Source: https://developers.google.com/ (latest access: April 2021)

Applications: topic classification

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

1 Introduction

Language model pre-training has been shown to be effective for improving many natural language processing tasks (Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018). These include sentence-level tasks such as natural language inference (Bowman et al., 2015; Williams et al., 2018) and paraphrasing (Dolan

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

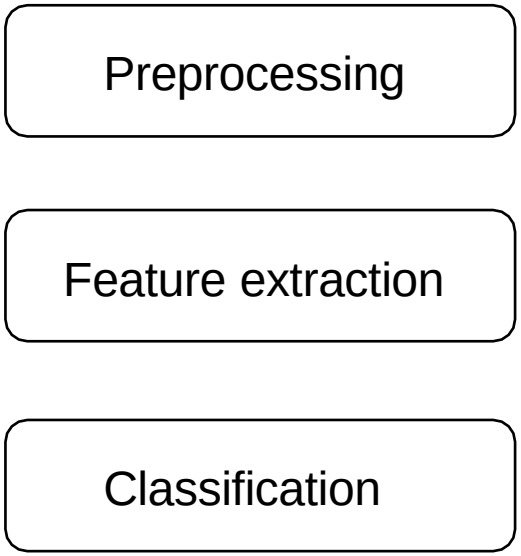
We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers of the Transformer (Vaswani et al., 2017). Such restrictions are sub-optimal for sentence-level tasks, and could be very harmful when applying fine-tuning based approaches to token-level tasks such as question answering, where it is crucial to incorporate context from both directions.

In this paper, we improve the fine-tuning based approaches by proposing BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.

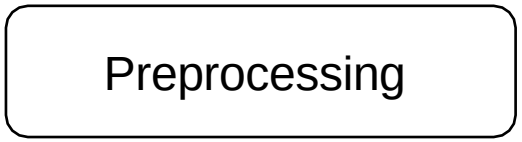
- **Computational Linguistics**
- Computer Vision
- Speech Synthesis
- Physics
- Quantitative Biology
- Quantitative Finance
- Economics
- …

Text categorization

- Input
  - A document d
  - A set of classes C = {c_1, c_2 , c_3 , …}
- Output
  - Predicted class $\mathbf{c_i} \in \mathbf{C}$



Preprocessing

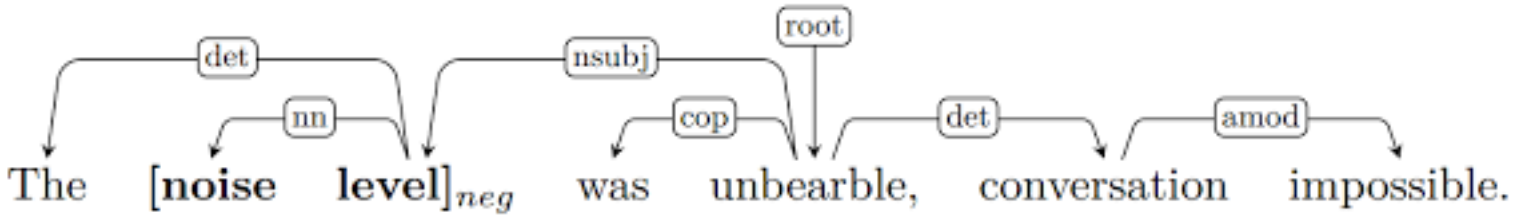


- Text is preprocessed for the subsequent steps
  - Sentence splitting
  - Word splitting
  - Lemmatization (tried -> try)
  - Stopwords removal (the, in, of ...)
  - ...

Preprocessing

Feature extraction

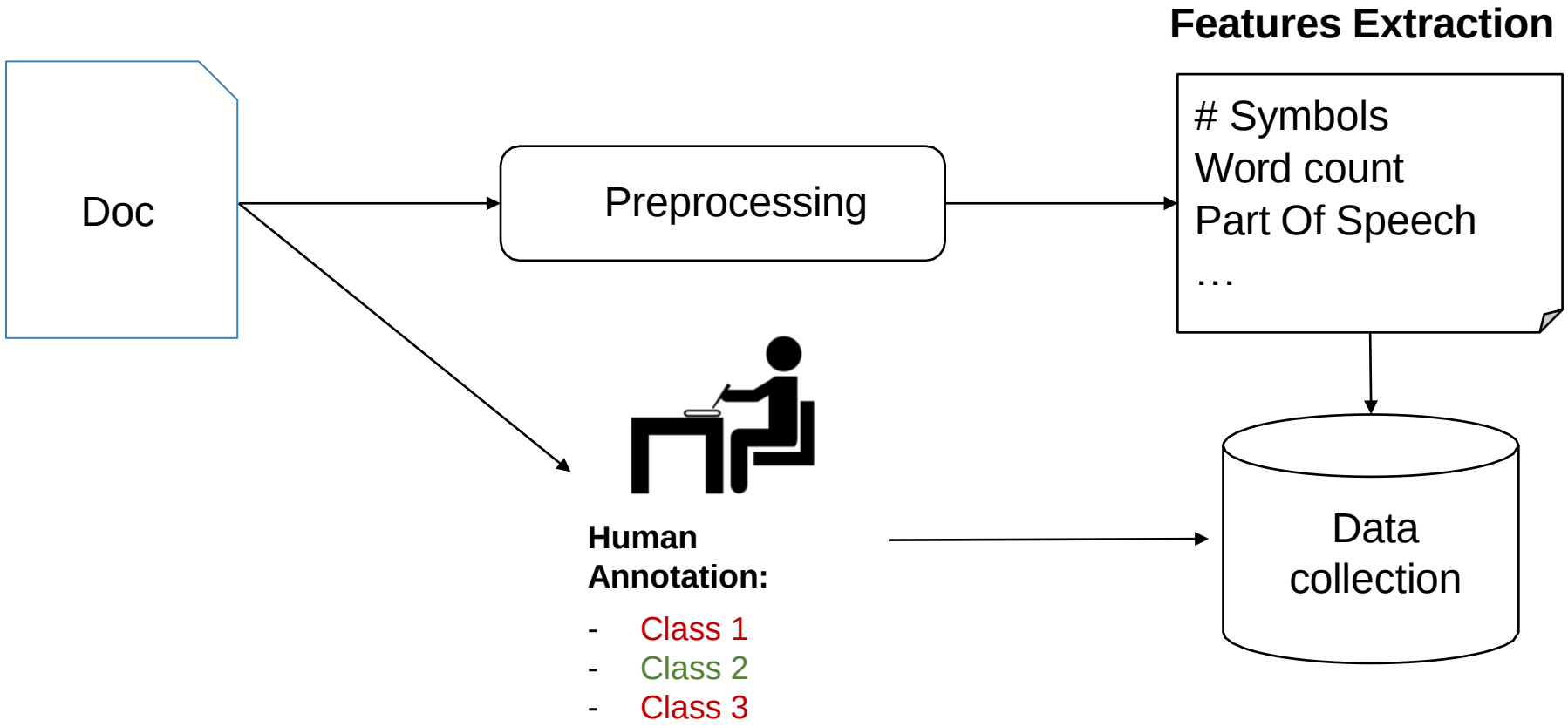
- Extract relevant features that can be used for the classification:
  - Part-Of-Speech tagging (try : VERB)
  - Dictionary-based word count
  - Relies on the Bag-Of-Word text representation
  - Semantic tree parsing (identification of word dependencies)
  - ...



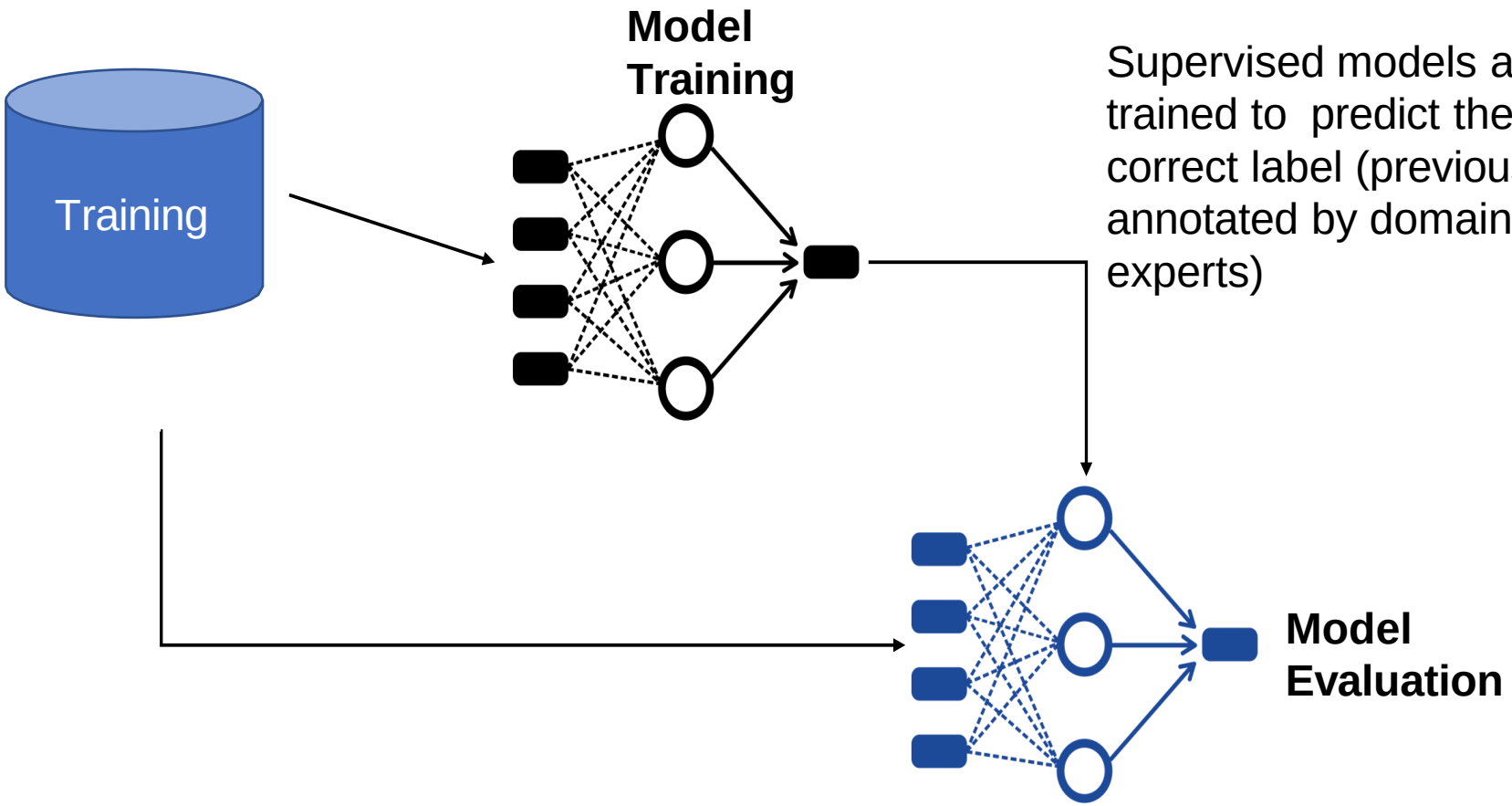
Rule-based classification

- Approach
  - Apply ad hoc rules based on either the presence of a combination of words or other textual features
- Example of application: spam detection
  - Relies on a blacklist of email addresses
    - Match the patterns «Get rich» and «you have been selected inside the email body
- Key aspects
  - Accuracy can be high (for domain-specific use cases)
  - Rules maintenance is costly and error-prone

Machine learning-based approach



Classification



Icon by David Christensen from the Noun Project

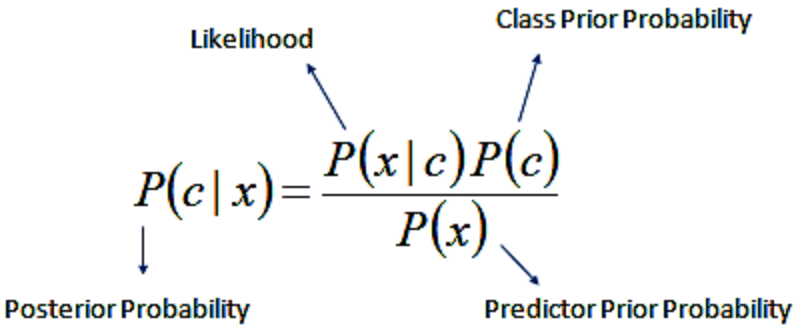
Supervised machine learning

Examples of classification algorithms commonly applied in text categorization

- **Naïve Bayes**
  - applies Bayes' theorem with naïve independence assumptions between the features
- **Logistic Regression**
  - use a logistic function to model the probability of a certain class
- **Support Vector Machines**
  - maps training examples to points in space so as to maximize the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.
- **K-Nearest Neighbors**
  - classifies a data point by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors

Supervised machine learning

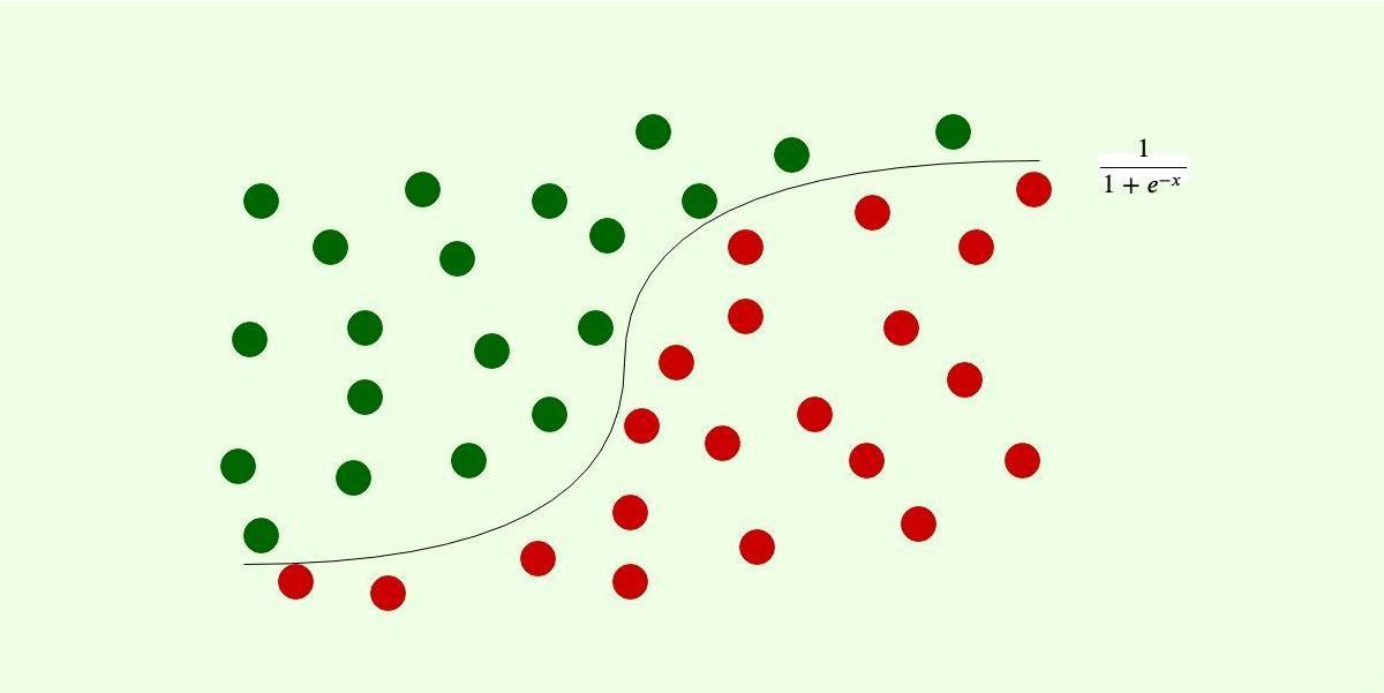
- Examples of classification algorithms commonly applied in text categorization
- Naïve Bayes
  - applies Bayes' theorem with naïve independence assumptions between the features



$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

Supervised machine learning

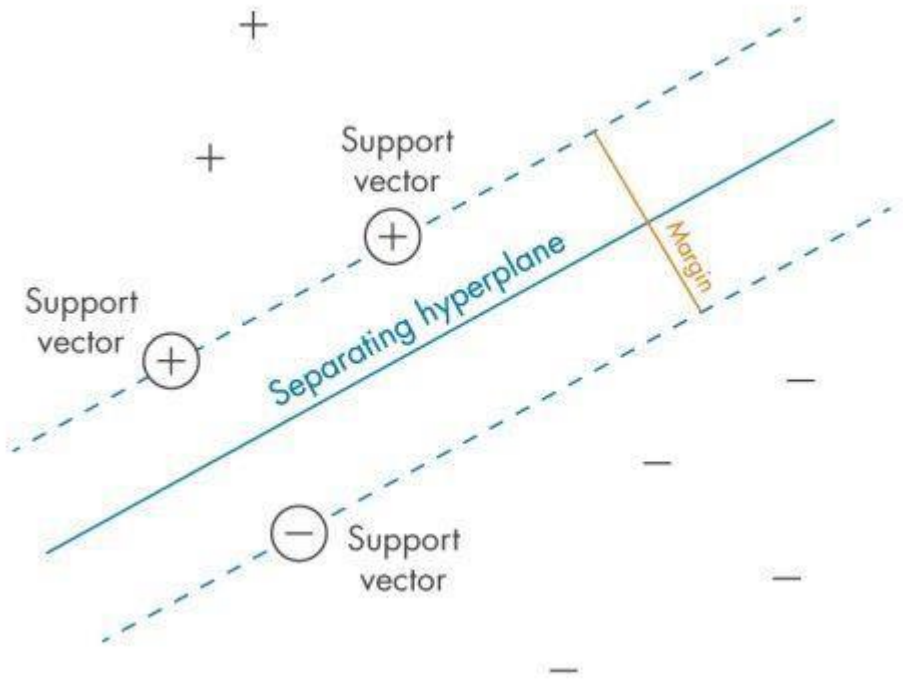
- Examples of classification algorithms commonly applied in text categorization
- Logistic regression



Source: https://it.mathworks.com (latest access: April 2021)

Supervised machine learning

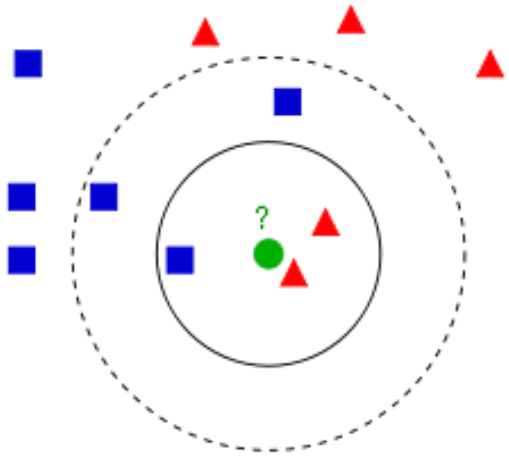
- Examples of classification algorithms commonly applied in text categorization
- Support Vector Machines



Source: https://it.mathworks.com (latest access: April 2021)

Supervised machine learning

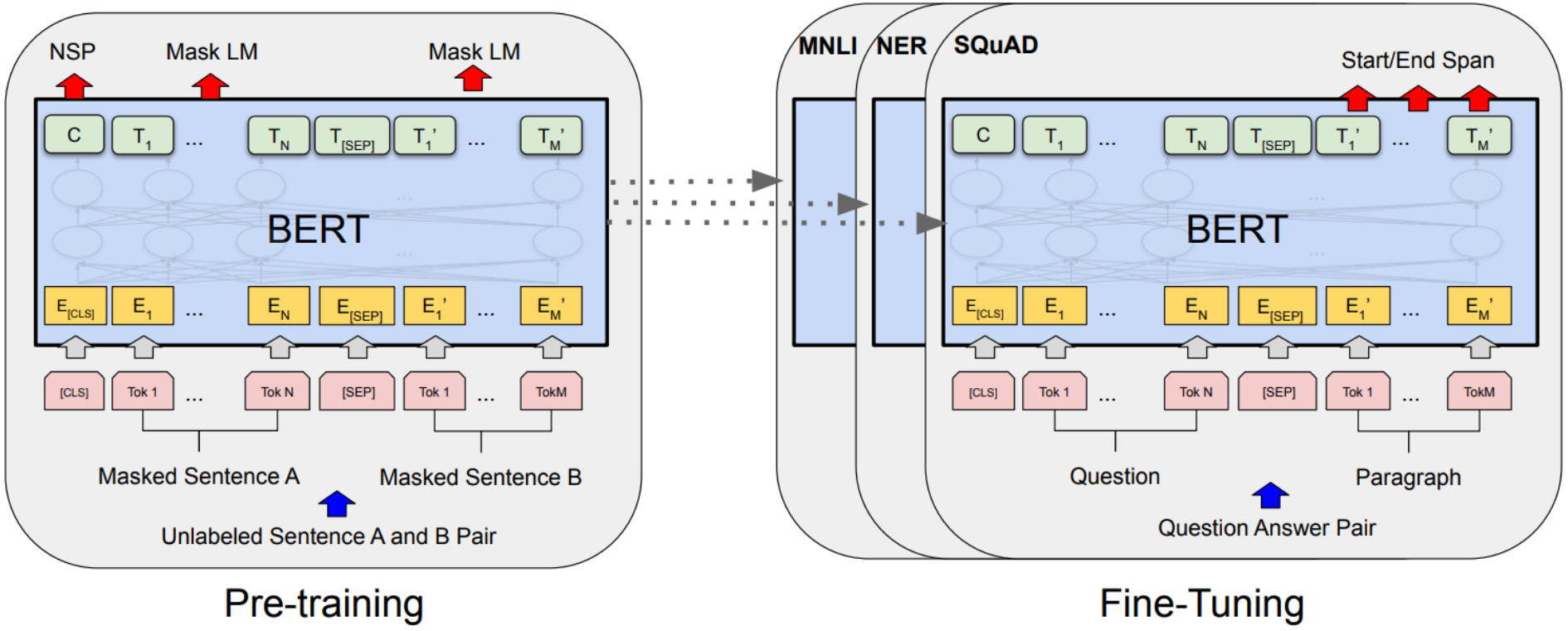
- Examples of classification algorithms commonly applied in text categorization
- K-Nearest Neighbor



Source: https://it.mathworks.com (latest access: April 2021)

Transformers for text categorization

- Self-supervised learning
  - Pretrained and fine-tuning



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova. Proceedings of NAACL-HLT 2019.

Evaluation metrics

- **Recall of class i**: Fraction of documents of class i that have been correctly classified

$$\frac{c_{ii}}{\sum_j c_{ij}}$$

- **Precision of class i**: Fraction of documents assigned to class that actually belong to i

$$\frac{c_{ii}}{\sum_j c_{ji}}$$

- **Accuracy:** Fraction of documents classified correctly (independently of the class)

$$\frac{\sum_i c_{ii}}{\sum_j \sum_i c_{ij}}$$

Notation used: $\mathbf{c_{ij}}$ is the number of documents of class i labeled as j

Model evaluation

- Confusion matrix
  - For each pair of classes <c1,c2> how many documents from c1 were incorrectly assigned to c2?

Documents (test set)	Sport	Business	Technology
Sport	**50**	10	9
Business	5	**80**	10
Technology	2	15	**98**

Model evaluation

- **Confusion matrix**
  - For each pair of classes **<c_1,c_2>** how many documents from c_1 were incorrectly assigned to c_2?

Documents (test set)	Sport	Business	Technology
Sport	**50**	10	9
Business	5	**80**	10
Technology	2	15	**98**

Predicted class

10 articles have incorrectly classified as technology instead of business

Actual class

Example

Documents (test set)	Sport	Business	Technology
Sport	**50**	10	9
Business	5	**80**	10
Technology	2	15	**98**

- Accuracy: Fraction of documents of class i that have been correctly classified

Accuracy = (50+80+98)/(50+10+9+5+80+10+2+15+98) = 228/279 = 81.7%

Example

Documents (test set)	Sport	Business	Technology
Sport	**50**	10	9
Business	5	**80**	10
Technology	2	15	**98**

- Recall of class Sport: Fraction of documents of class Sport that have been correctly classified

$$\text{Recall} = (50)/(50+5+2) = 50/57 = 87\%$$

Example

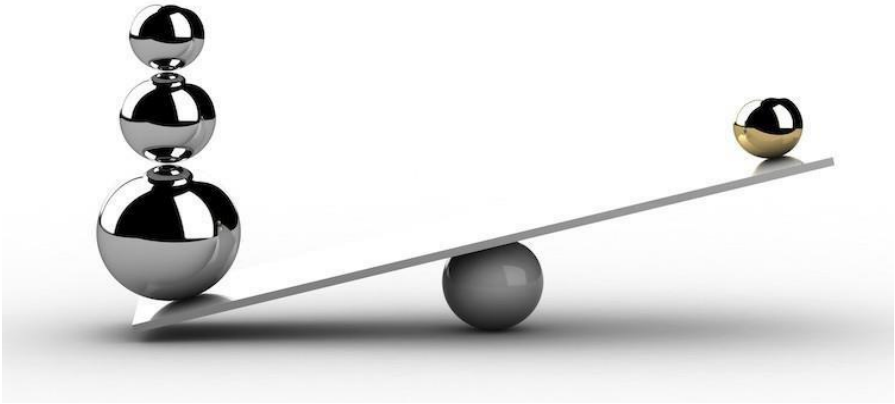
Documents (test set)	Sport	Business	Technology
Sport	**50**	10	9
Business	5	**80**	10
Technology	2	15	**98**

- Precision of class Sport: Fraction of documents assigned to class Sport that actually belong to Sport

$$\text{Precision} = (50)/(50+10+9) = 50/69 = 72.4\%$$

Challenges

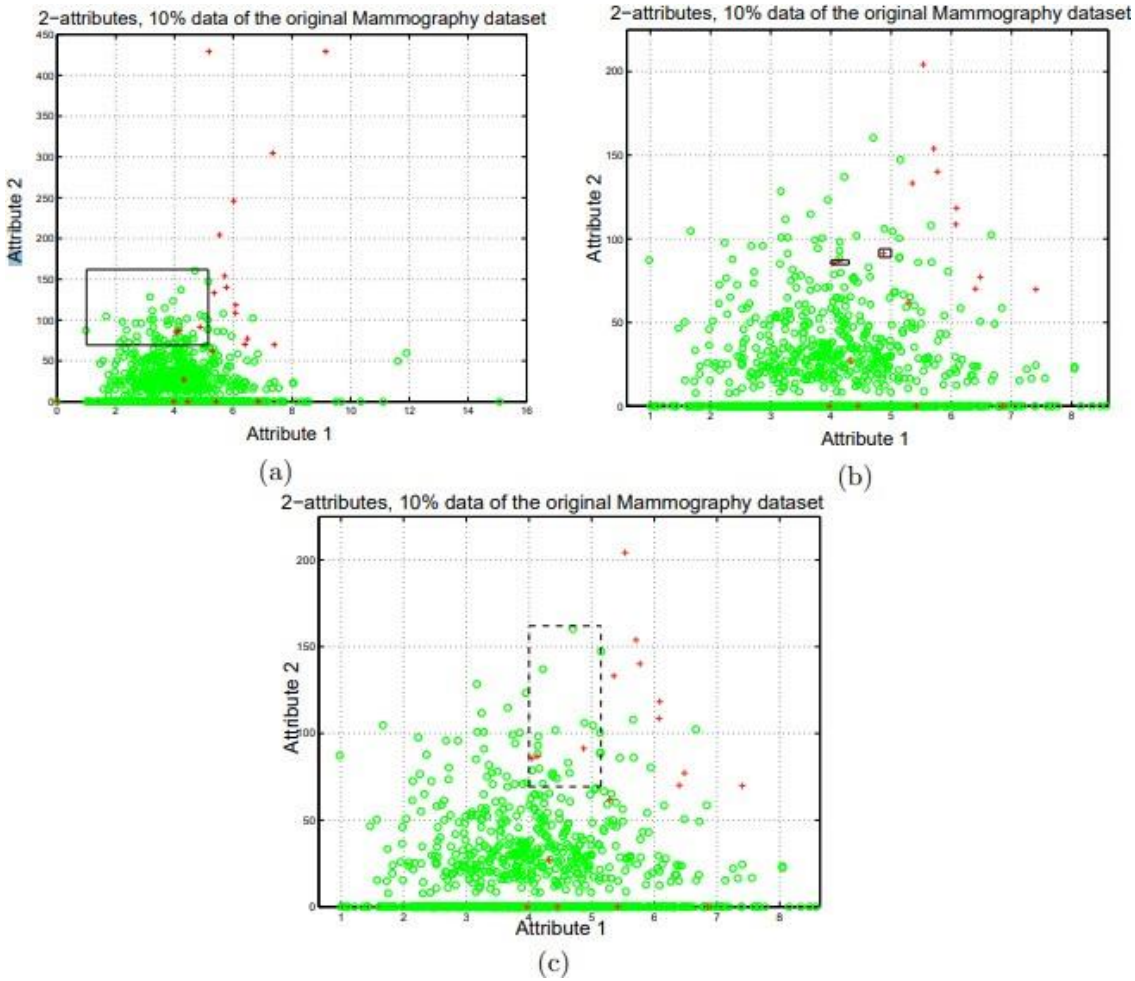
- Data imbalance
  - Input samples in the training data are not equally distributed among classes
- Solutions
  - Oversampling of the minority class
    - E.g., Smote [Chawla 2002]
  - Undersampling of the majority class



Source: https://towardsdatascience.com/ (latest access: April 2021)

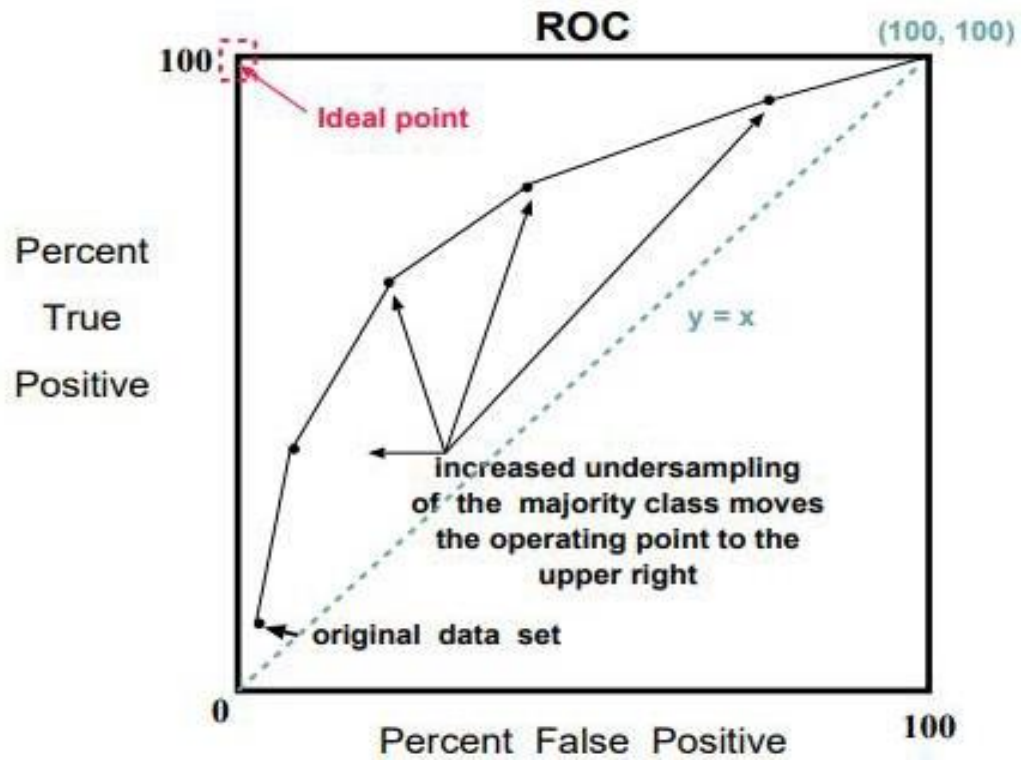
[Chawla 2002] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. J. Artif. Int. Res. 16, 1 (January 2002), 321–357.

Challenges



[Chawla 2002] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. J. Artif. Int. Res. 16, 1 (January 2002), 321–357.

Challenges



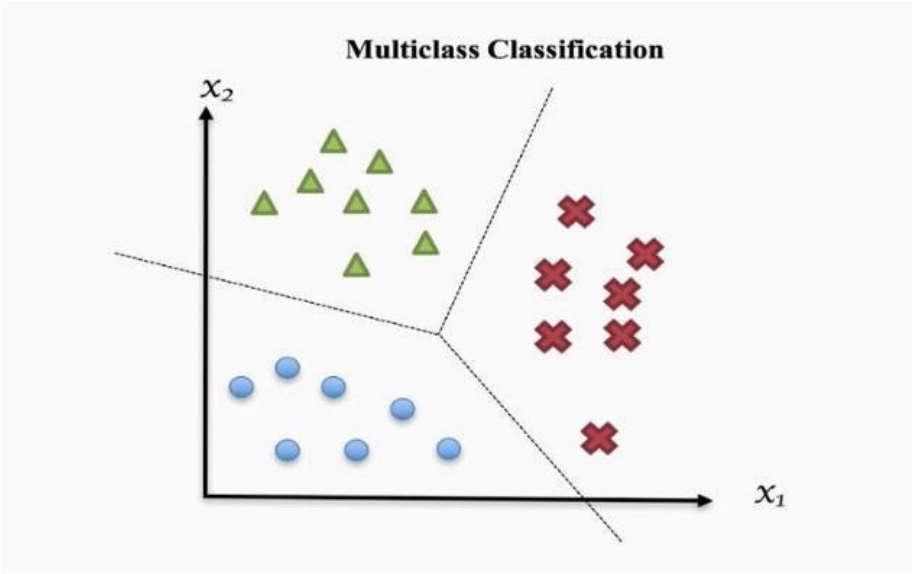
[Chawla 2002] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002.
SMOTE: synthetic minority over-sampling technique.
J. Artif. Int. Res. 16, 1 (January 2002), 321–357.

Challenges

- **Multiclass text classification**
  - Classify text into three or more classes

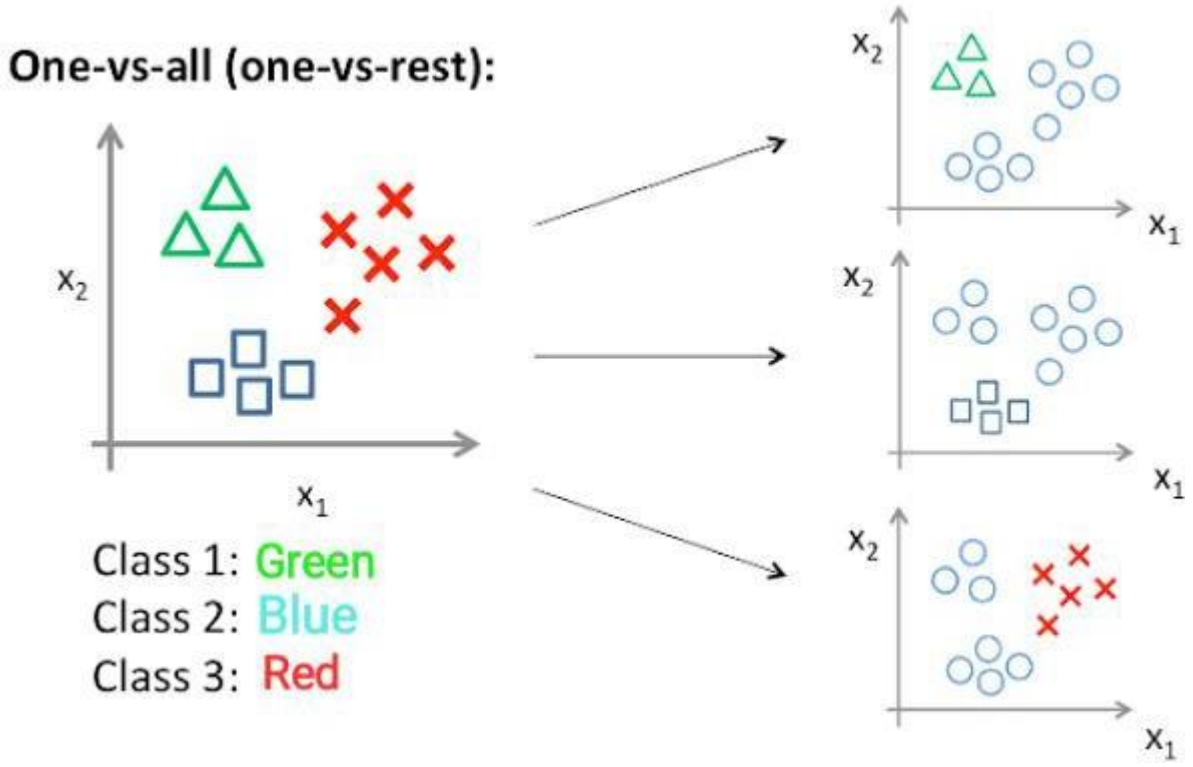
- **Examples of applications**
  - Automated ticketing
  - Taxonomy generation

- **Solution**
  - One vs. all



Source: https://towardsdatascience.com/ (latest access: April 2021)

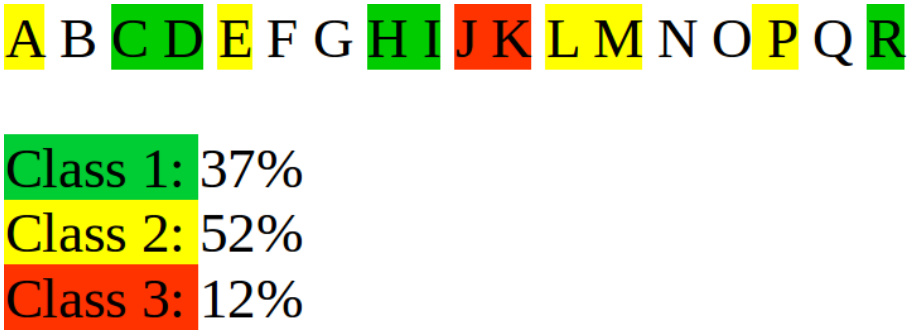
Challenges



Source: https://towardsdatascience.com/ (latest access: April 2021)

Challenges

- **Multilabel text classification**
  - Multiple labels may be assigned to each text
  - **Examples of applications**
    - Automated ticketing
  - Taxonomy generation
- **Solution**
  - Binary classification
    - Neglects class relationships
  - Ad hoc solutions



Source: https://towardsdatascience.com/ (latest access: April 2021)

Challenges

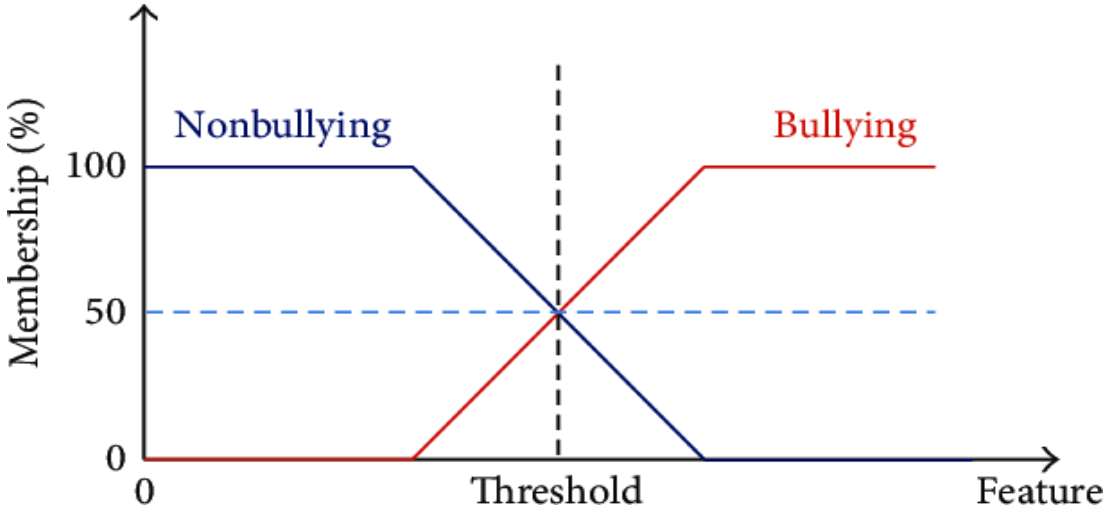
Fuzzy text classification

- Assign a probability to each class

- **Examples of applications**
  - Automated ticketing
  - Taxonomy generation

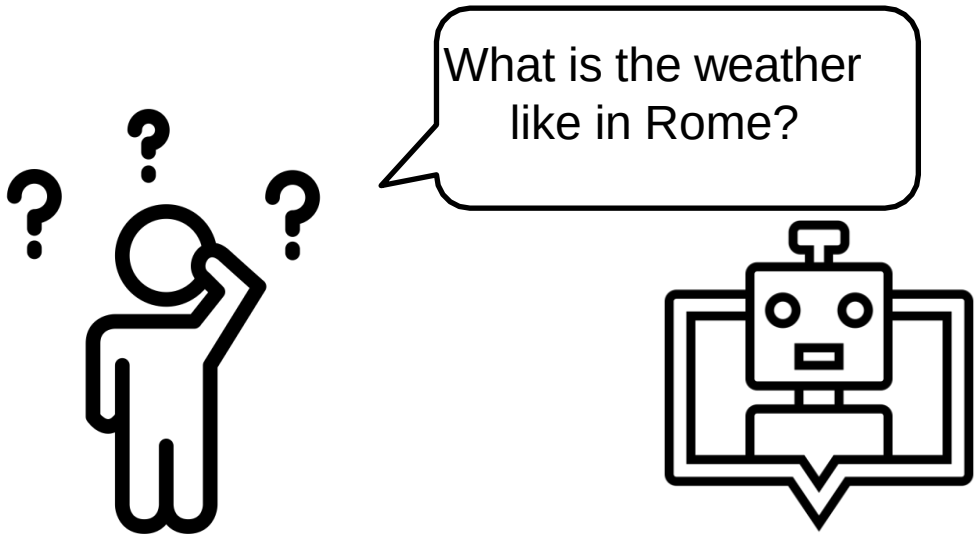
Solution

- Fuzzy logic
- Neural network models



Source: https://towardsdatascience.com/ (latest access: April 2021)

Intent detection



Icons credit: Creative Mania & Rahmat Dwi Cahyo from the Noun Project

Intent detection

1 NER module → **Rome**: LOC

2 NED module → **Rome**: /wiki/Rome

- The NER and NED modules are used to understand the entities involved
- The system is unaware of the action users are willing to take

Intent detection

1. NER module → **Rome**: LOC
2. NED module → **Rome**: /wiki/Rome
3. Intent Detection module → **Intent**: - get_weather; **Entity**: - /wiki/Rome

Icons credit: Creative Mania & Rahmat Dwi Cahyo from the Noun Project

Intent detection

- **Intent definition**
  - «the fact that you want and plan to do something» (Cambridge Dictionary)
- **Intent Detection** is a core element of any task-oriented conversational system.

Image credit: https://tangowork.com

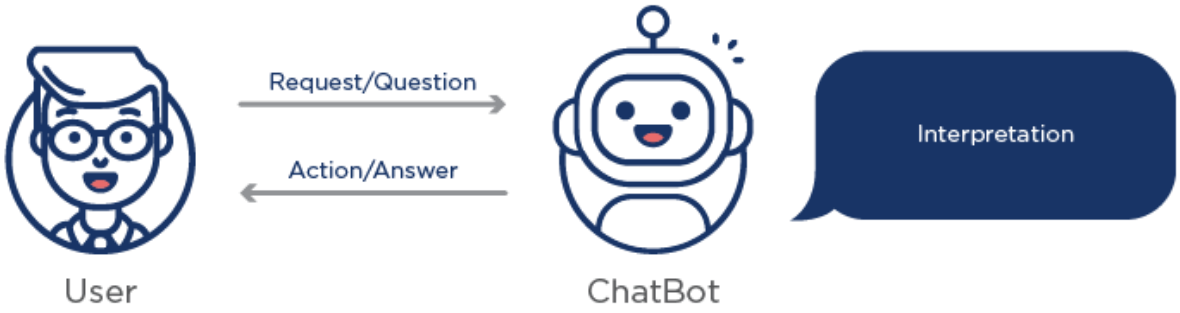
Intent detection: use cases

- **Online marketing**
  - recognize user intentions in puchasing goods or services
- **Customer care**
  - shunt users to specific company's divisions
  - avoid customer churns
- **Product launch analysis**
  - Predict the success or failure of a given product
  - It usually combines sentiment analysis with intent recognition
- **Track user patterns**
  - E.g., track user quit intentions as consequence of the Netflix pricing modification on 2011

Intention Analysis - Business Use Cases - http://www.aiaioo.com (latest access: June 2021)

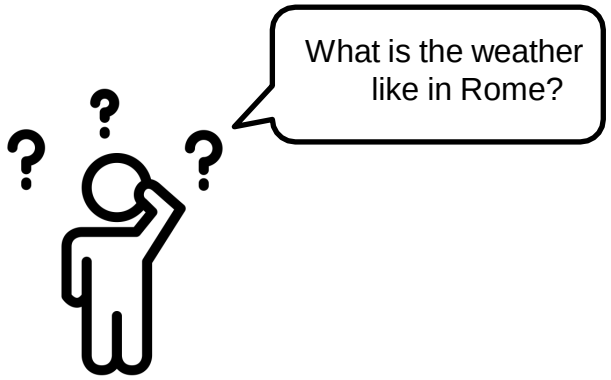
What is an intent?

- It is formulated as a classification problem
  1. Define a set of possible intents
  2. Analyze the user-provided query
  3. Assign the most likely class (predefined intents)



Intent classes

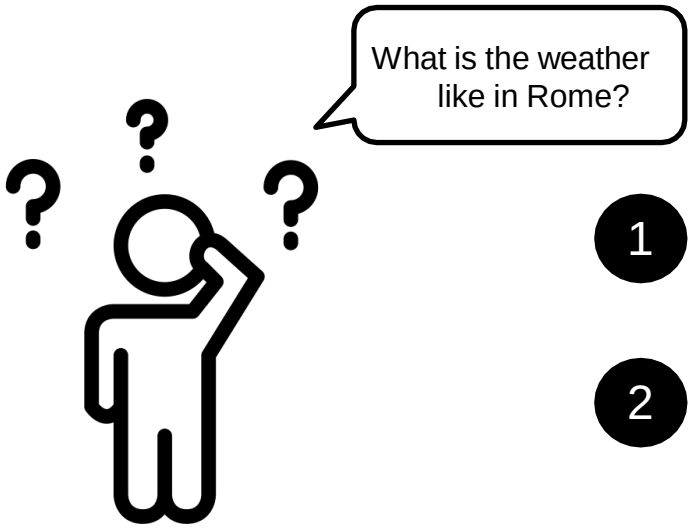
- Intents represent a task the conversational agent needs to perform
- The definition of intent classes depends on the application domain
  - Finance: new_buy, get_spent, pursue_total
  - Music: add_to_playlist, play_song, play_composer
  - Insurance: get_info, find_doctor, claim_procedure, verify_coverage ...



Icons credit: Rahmat Dwi Cahyo from the Noun Project

Intent detection

- User **queries** are processed to
  - Get the entities involved in the interaction
  - Get the semantic information related to the query
  - Convert the query to machine readable format

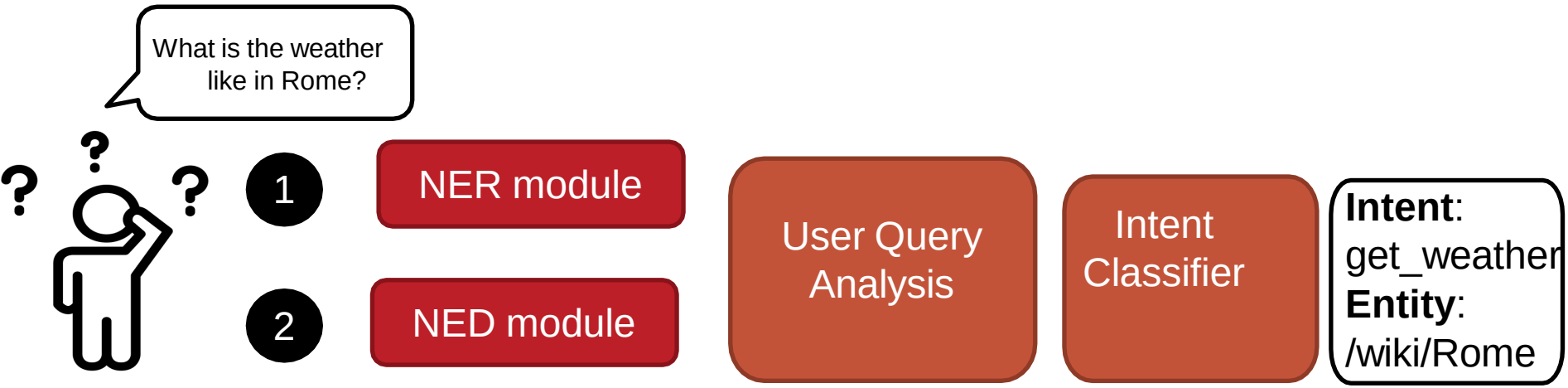


1. NER module
2. NED module

User Query Analysis

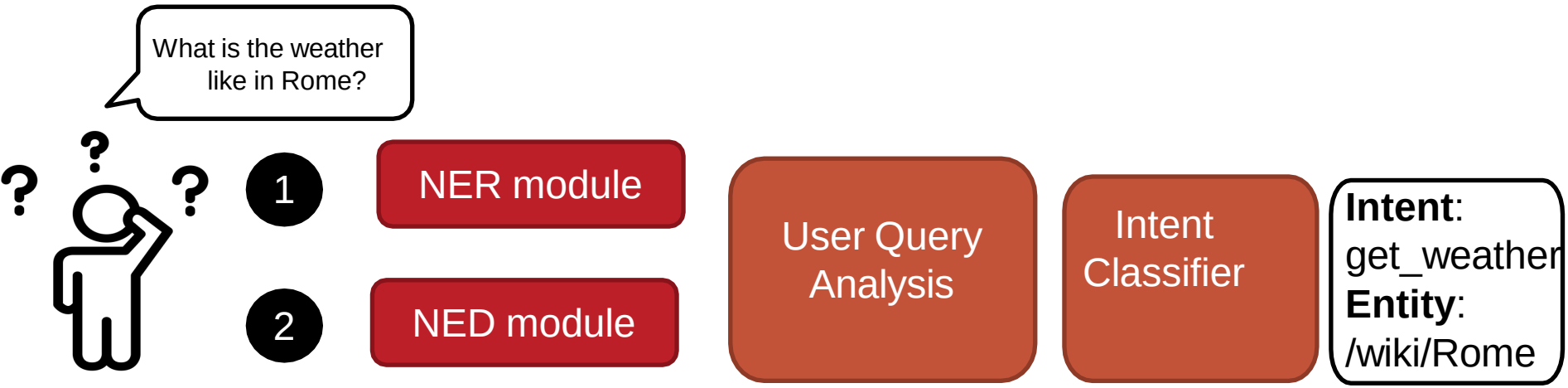
Intent classification

- Leveraging the information obtained during the analysis, classify the intent in one of the defined classes



Intent classification

- Leveraging the information obtained during the analysis, classify the intent in one of the predefined classes
- Methods
- Regular expressions
- Machine Learning classifiers
- Deep learning models



Regular expressions

Advantages

Simpler than machine learning approaches

Highly computationally efficient

Drawbacks

Nested conditions difficult to maintain

Low generalization capabilities

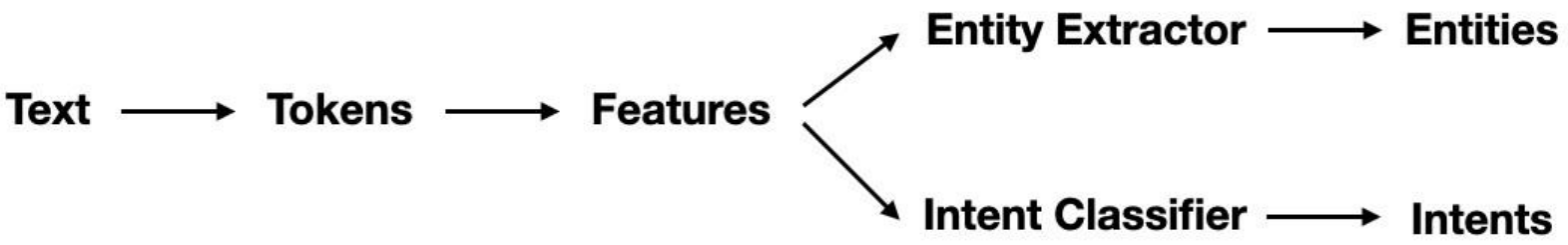
```
if re.search("playlist", user_query) and re.search("add", user_query):
    # add to playlist
elif re.search("playlist", user_query) and re.search("remove", user_query):
    # remove from playlist
```

✓ Add the song Sound of Silence to my playlist

✗ Include Sound of Silence to my favourite songs

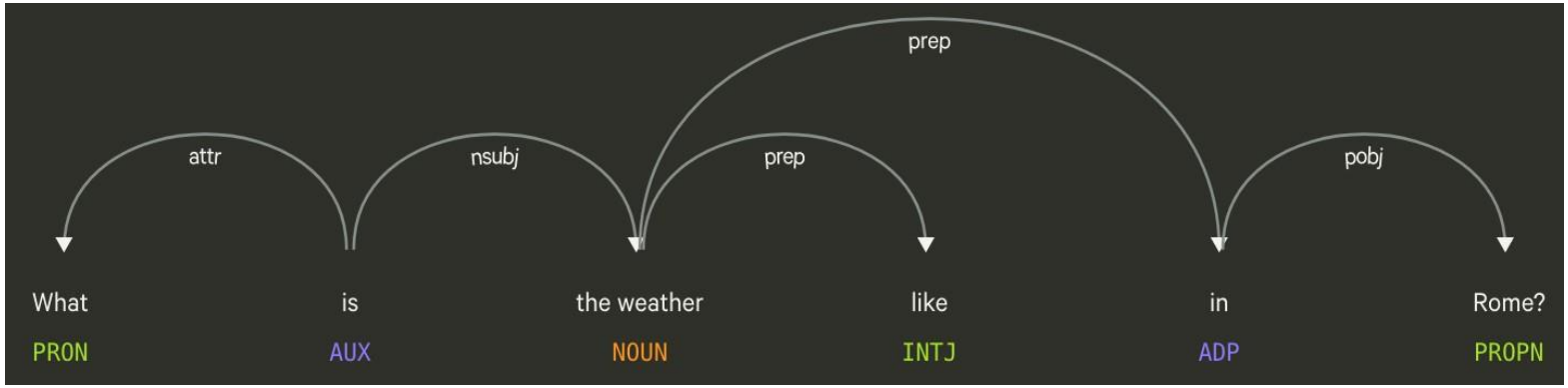
Machine learning models

- They exploit the same preprocessing pipeline of NER/NED modules
- The intent is often linked to the type of entities involved in the query



Machine learning models

- **Key features** for intent classification
  - **Verbs** are usually the most relevant syntax units
  - **Dependency parsing** tree is relevant to understand the intent

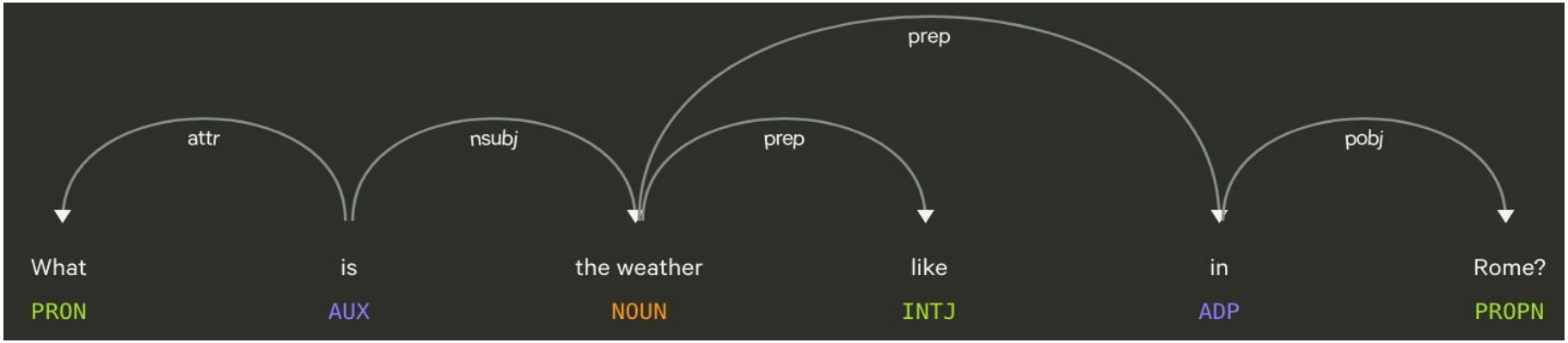


Output of Spacy dependency visualizer. https://spacy.io

Dependency tree

Dependency trees are used to identify the relations between tokens

- **nsubj:** nominal subject
- **pobj:** object of a preposition
- **attr:** complement of a copular verb
- **prep:** prepositional modifier
- Ouput of Spacy dependency visualizer

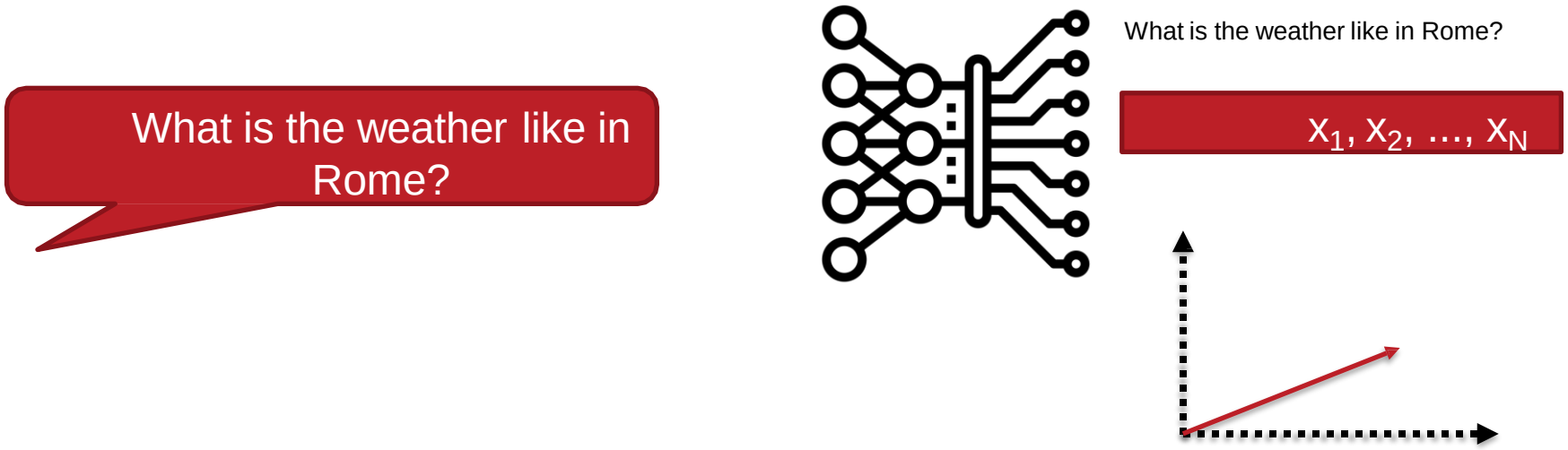


Output of Spacy dependency visualizer. https://spacy.io

De Marneffe, M. C., & Manning, C. D. (2008). *Stanford typed dependencies manual* (pp. 338-345). Technical report, Stanford University.

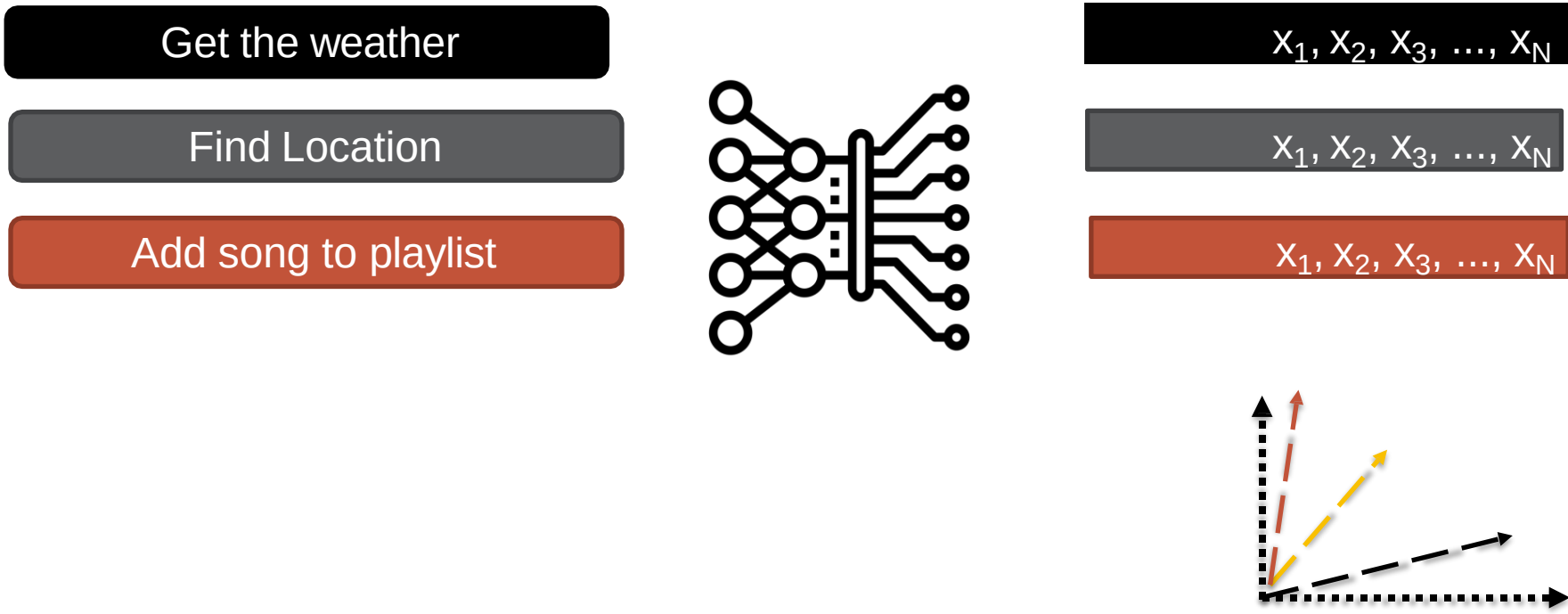
Deep learning representations

- Deep language models are able to analyze raw text with minimal preprocessing
- A sentence is encoded into a high-dimensional vector representation that embed semantic information



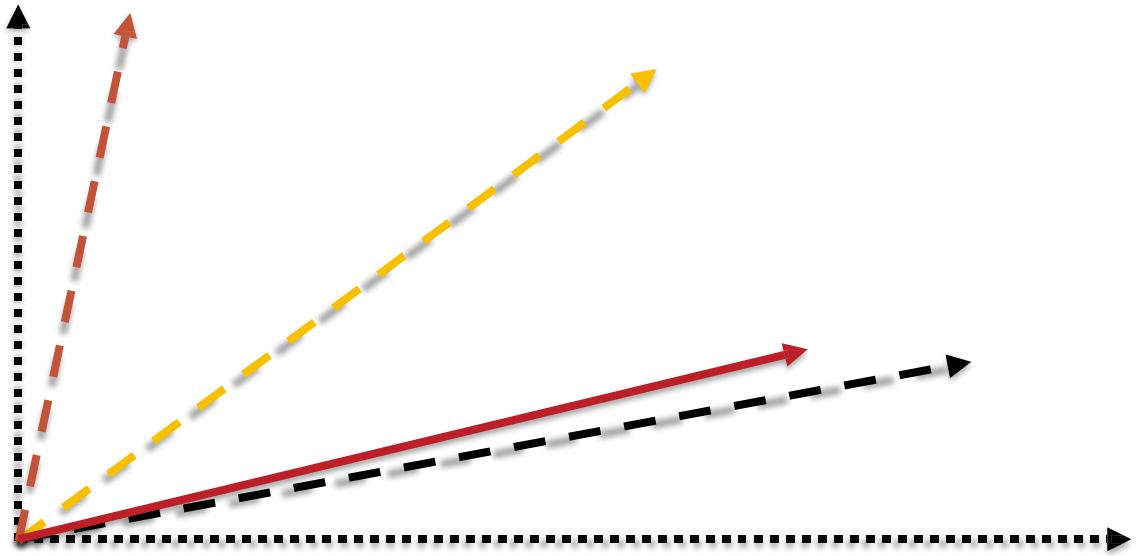
Deep Learning representations

- Class-representative sentences are also encoded into a the latent space
  - They capture semantic text similarities



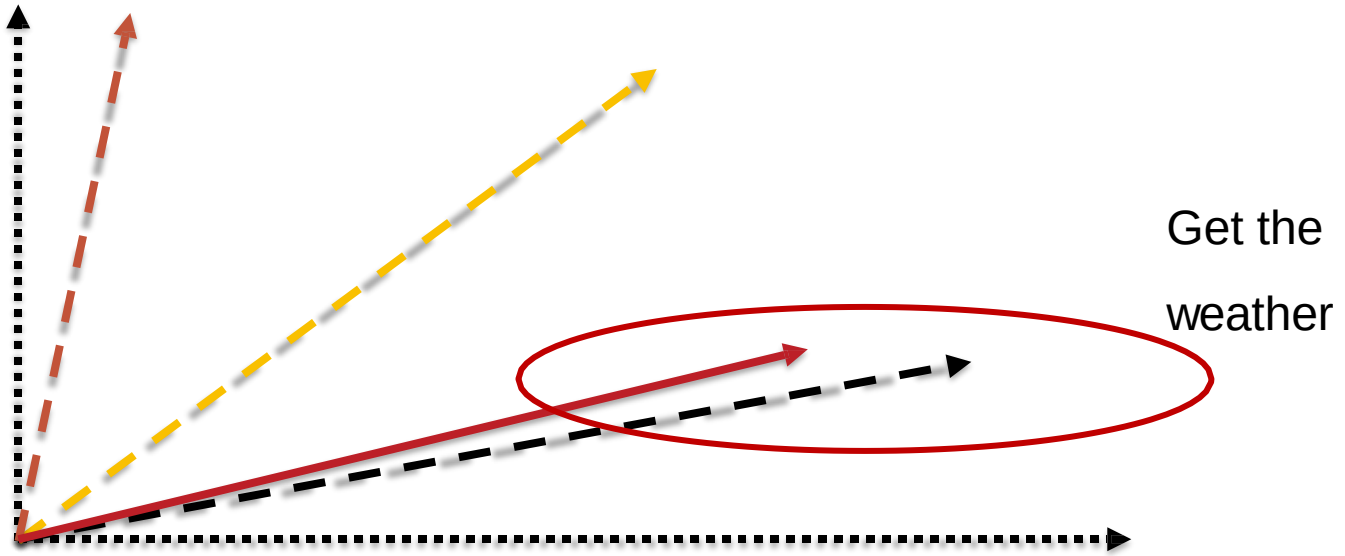
Deep Learning representations

- **Intent classification**
  - Assign the intent whose latent representation is most similar to the user query



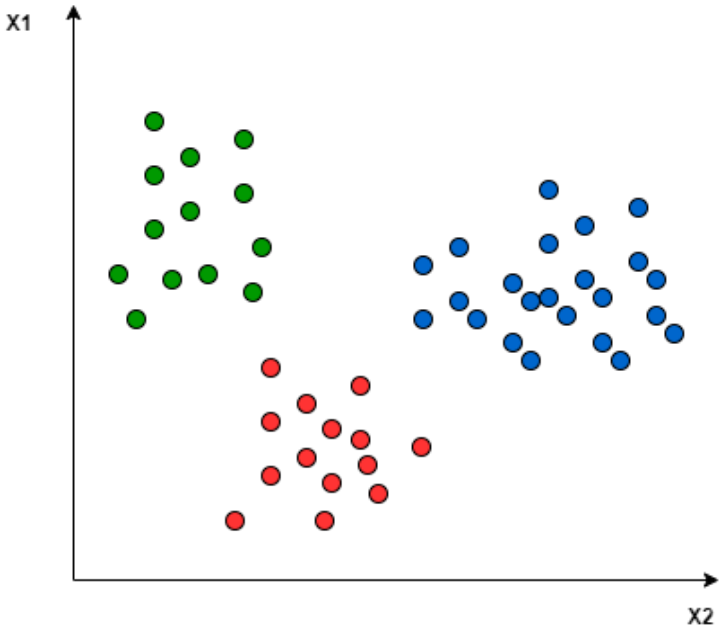
Deep Learning representations

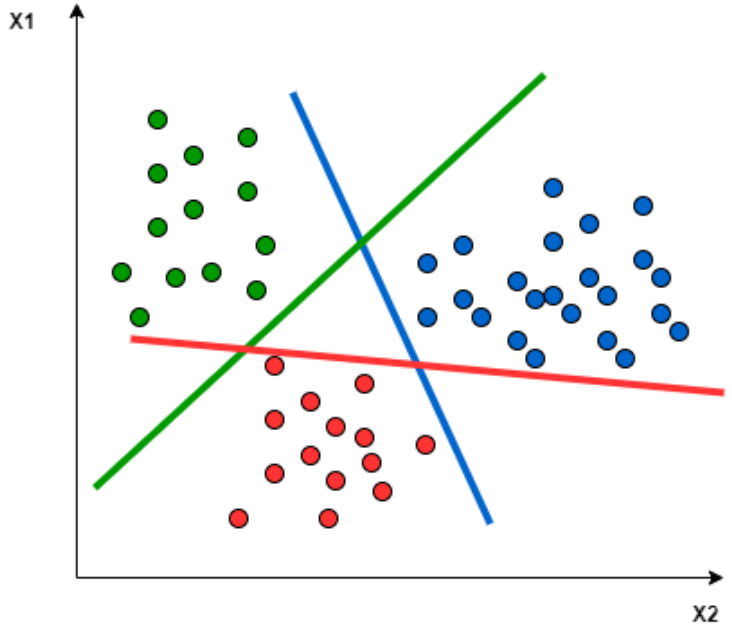
- **Intent classification**
  - Assign the intent whose latent representation is most similar to the user query



Machine learning classification on top of Deep Learning representations

- **Approaches**
  - **Unsupervised** approach
    - It does not require annotated data
    - Based on **pretrained** models
  - **Supervised** machine learning algorithms
    - Trained on top of semantic deep learning representations of text





SVM example by: https://www.baeldung.com/cs/svm-multiclass-classification

Multi-facet query formulation

Humans tend to represent the intent using different formulations

Semantic analysis and natural language understanding are required.

Check IT ticket status

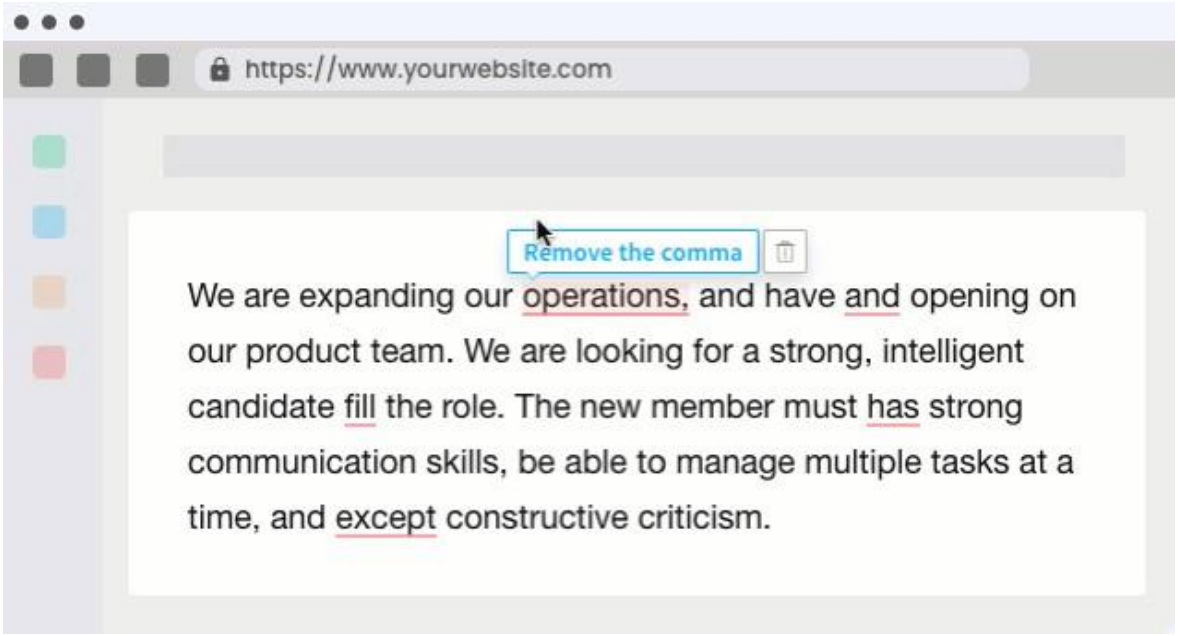
User Intent

- What is the status of my tickets?
- Is my issue fixed yet?
- What is the status of my request?

User input queries

Typos or wrong transcriptions

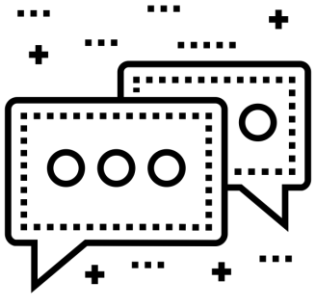
- Spell checking and syntax verification is required before the text processing step



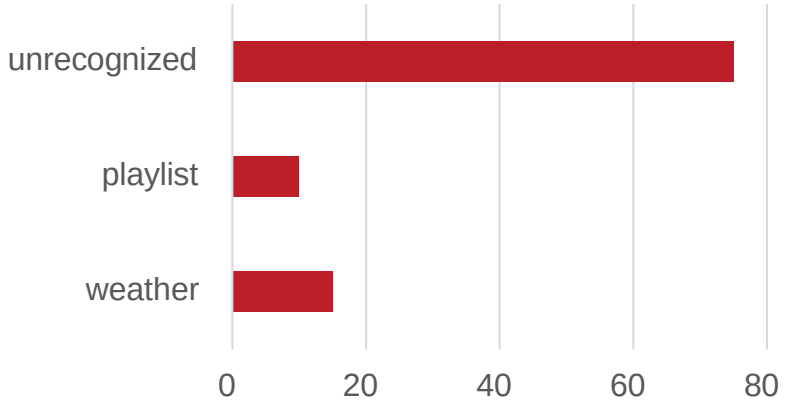
https://www.perfecttense.com/

Limited set of intents

- The set of possible classes could limit open domain intent detection



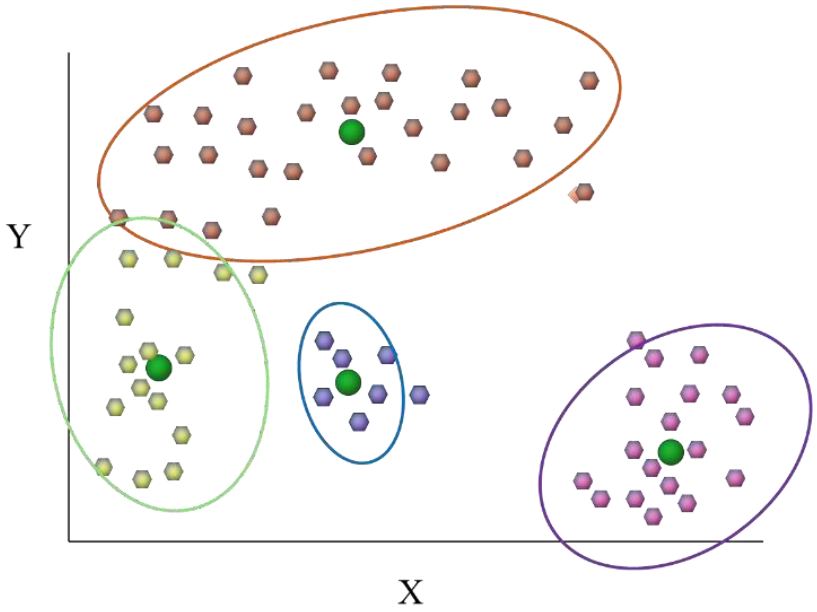
Intent Detection module



Icons credit: ProSymbols from the Noun Project

Limited set of intents

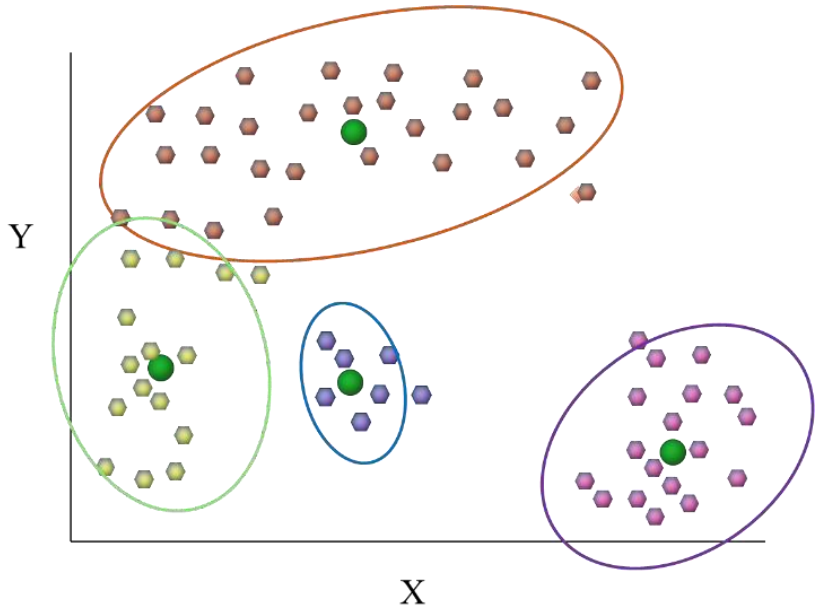
- Monitor user activities to find possible new intents
  - Unsupervised analytics (clustering)
  - Human invention



- open_app
- book_restaurant
- web_search
- call_contact

Limited set of intents

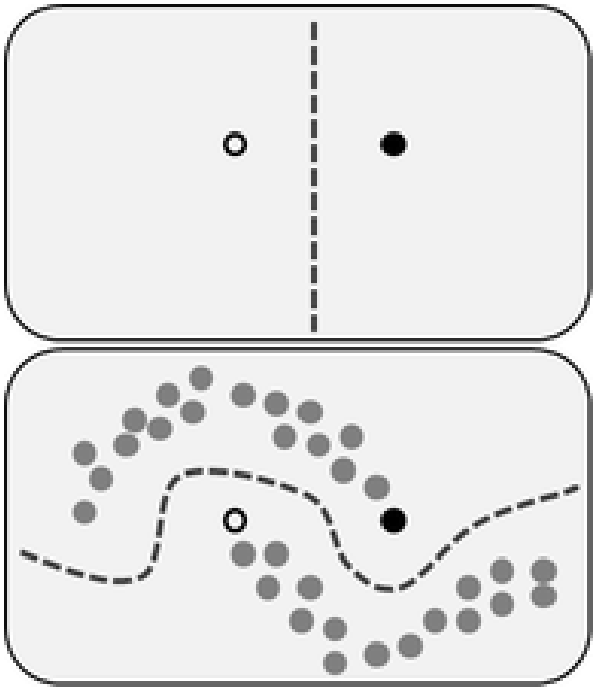
- The user experience could benefit from new classes



- open_app
- book_restaurant
- web_search
- call_contact

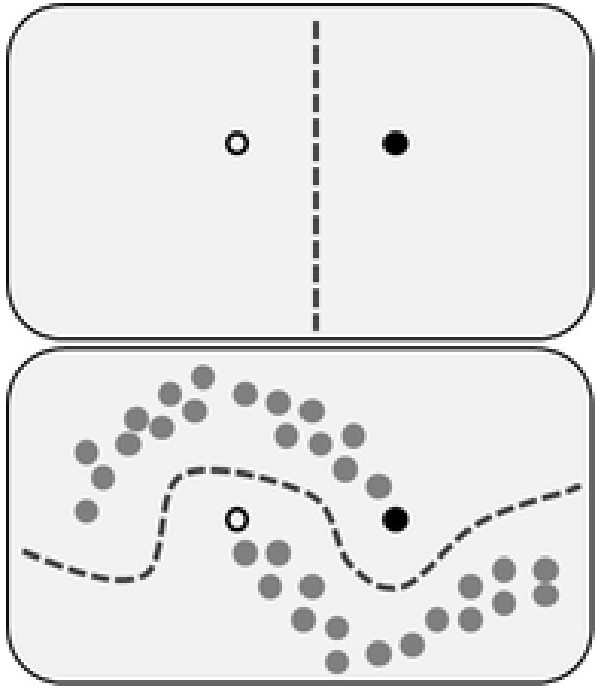
Semi-supervised learning

- Combine a small amount of labeled data with a large amount of unlabeled data during training
- The decision boundary can be adapted to the distribution of unlabeled data as well



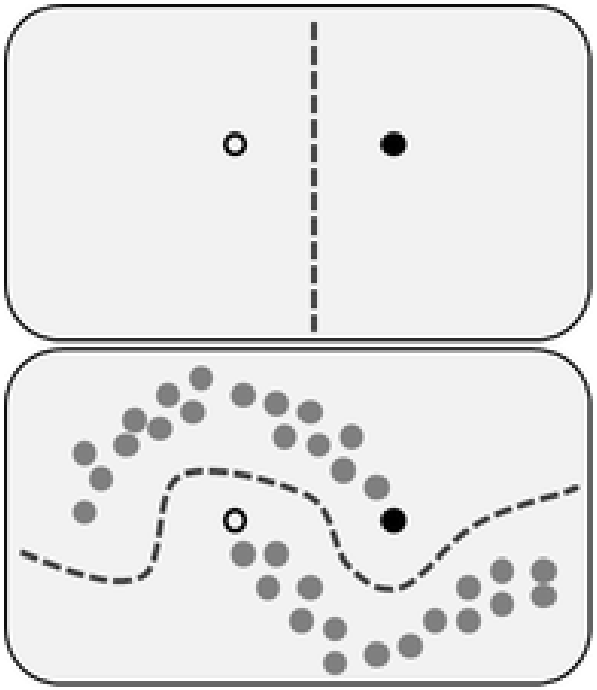
Semi-supervised learning

- Continuity assumption
  - Points that are close to each other are more likely to share a label
- Cluster assumption
  - The data tend to form discrete clusters
  - Points in the same cluster are more likely to share a label
- Manifold assumption
  - The data lie approximately on a manifold of much lower dimension than the input space



Semi-supervised learning

- Analyze user queries
- Extend intent assignments
- Identify new intents
- Correlate existing intents



Chatbots

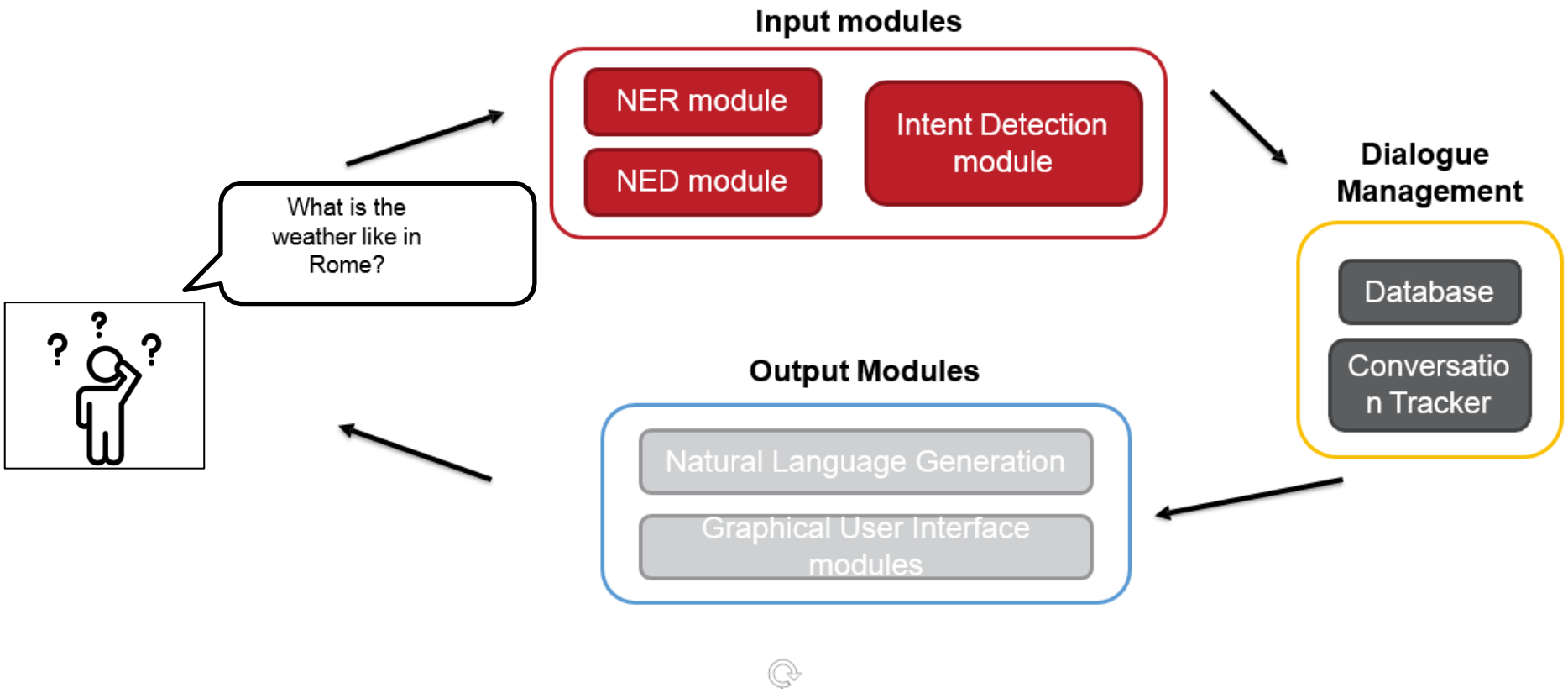
- A chatbot is a software application used to conduct an **online chat conversation** via text or text-to-speech
- It provides direct contact with a live human agent

Image: https://www.instilla.it
Text: Wikipedia: https://en.wikipedia.org/wiki/Chatbot

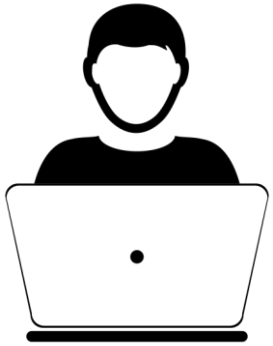
Chatbot applications

- Service profiling
  - Offer personalized help to improve customer experience
- Improve engagement on online users
  - automate user interaction on websites to engage and get useful feedbacks
- Conversational AI
  - Make user-Webapp interactions easier
- Recruiting
  - Manage large candidate pools
  - Speed up preliminary interviews

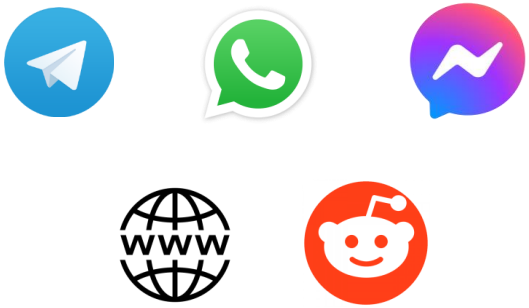
Chatbots: the pipeline



Chatbot platforms



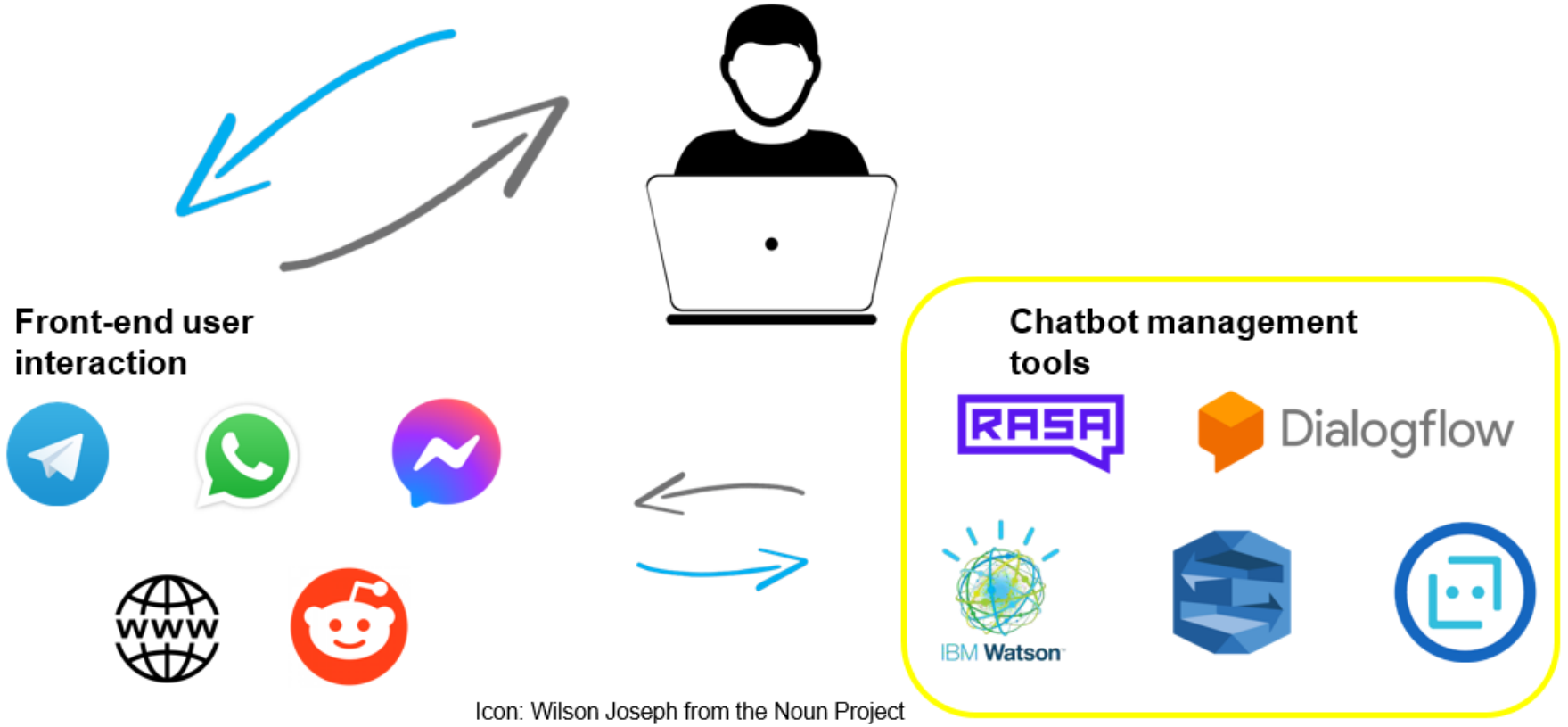
Front-end user interaction



Chatbot management tools

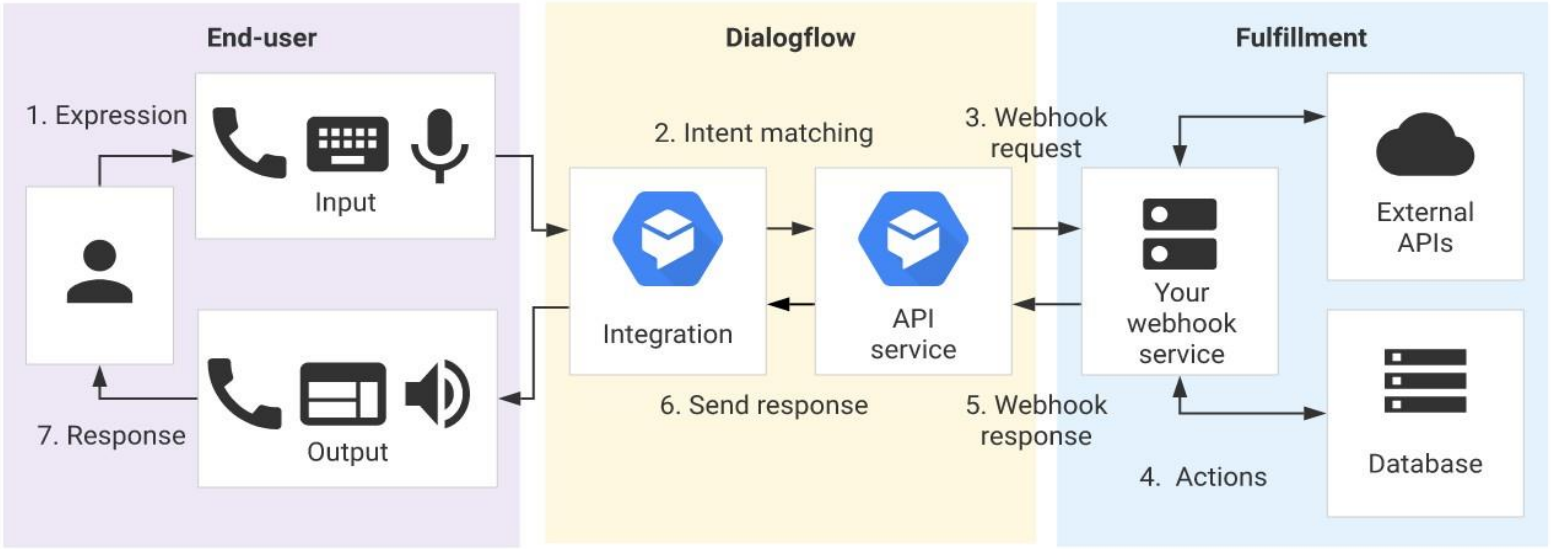
Icon: Wilson Joseph from the Noun Project

Chatbot platforms



Chatbot development and management tools

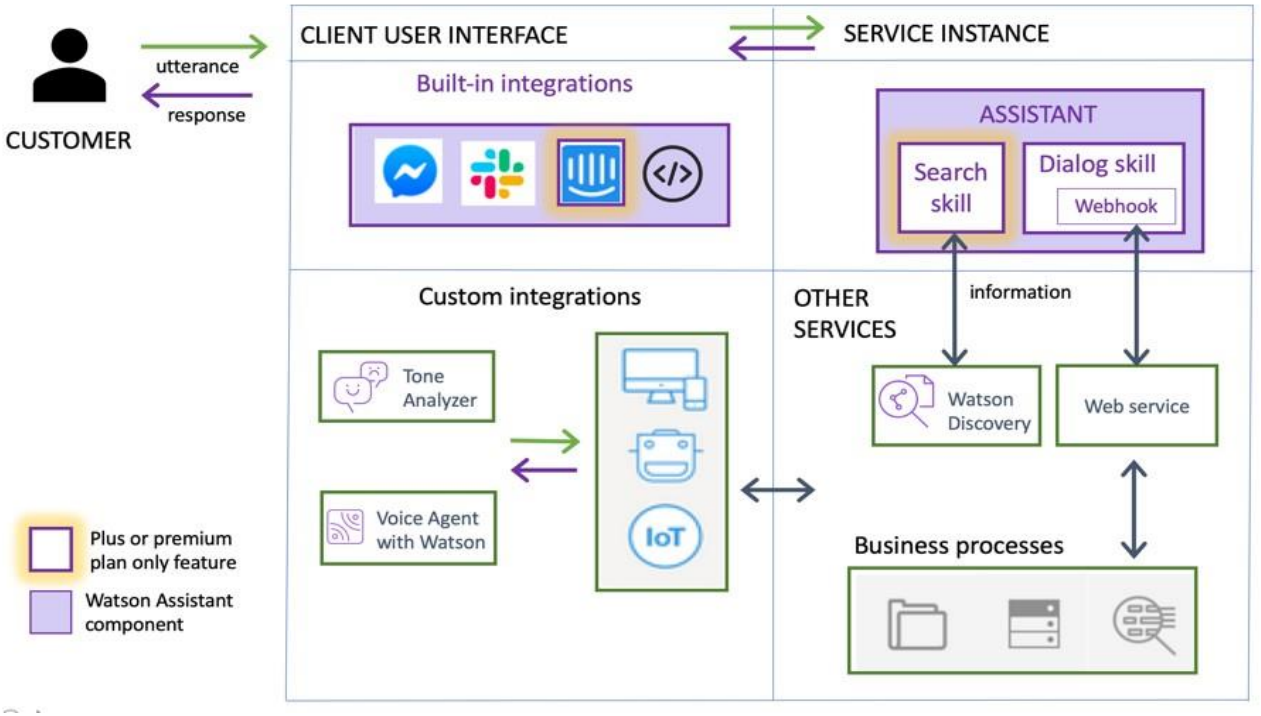
Dialogflow



https://cloud.google.com/dialogflow/ (latest access: April 2021)

Chatbot development and management tools

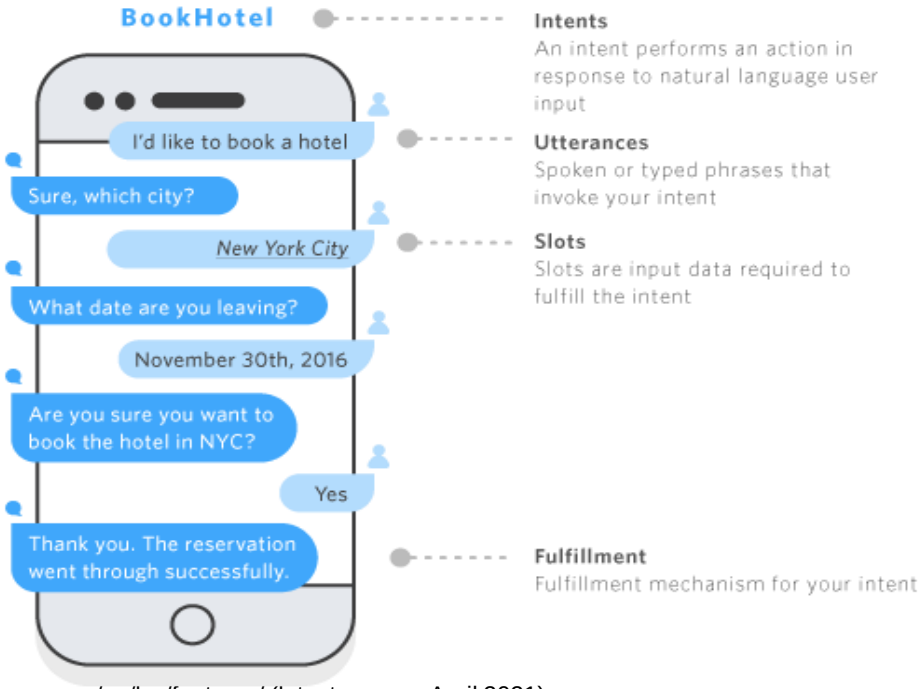
IBM Watson



https://developer.ibm.com/technologies/artificial-intelligence/articles/introduction-watson-assistant/ (latest access: April 2021)

Chatbot development and management tools

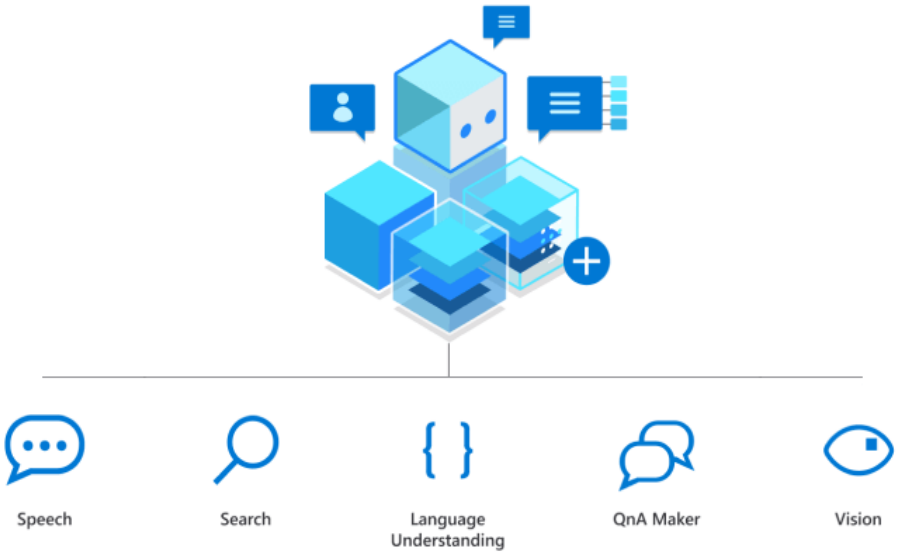
Amazon chatbot development framework



https://aws.amazon.com/en/lex/features/ (latest access: April 2021)

Chatbot development and management tools

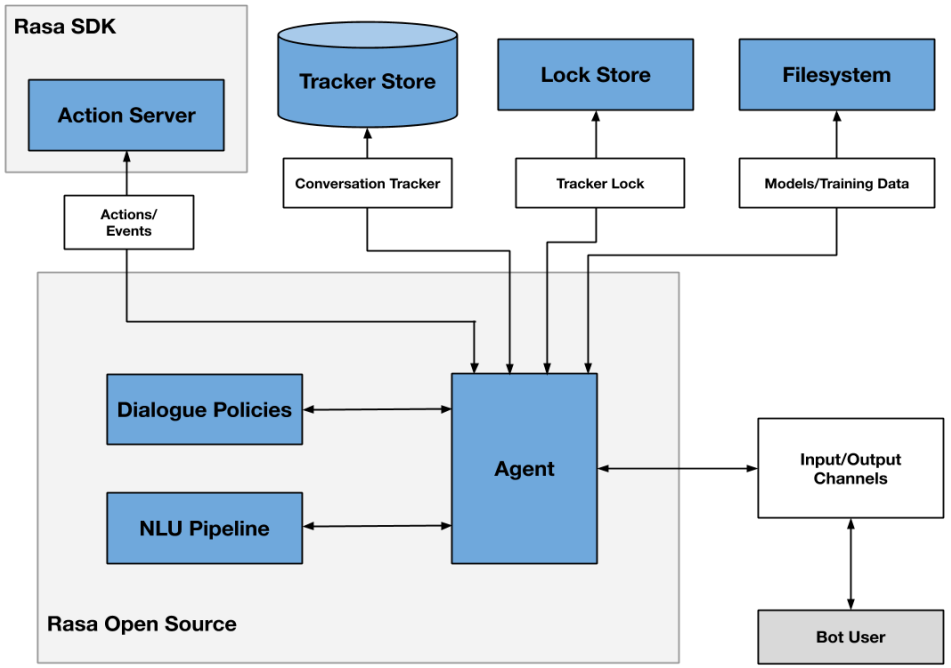
Microsoft Azure



https://azure.microsoft.com/it-it/services/bot-services/ (latest access: April 2021)

Chatbot development and management tools

RASA NLU



https://rasa.com/docs/rasa/arch-overview/ (latest access: April 2021)

Tool comparison

Platform	Documentation	Configuration	Testing	Pricing
Google Dialog Flow	★★★★☆	★★★★☆	★★★★☆	★★★☆☆
Amazon Lex	★★★☆☆	★★★★☆	★★★★☆	★★★★☆
IBM Watson	★★★★☆	★★★★★	★★★★☆	★☆☆☆☆
Microsoft Azure	★★★☆☆	★★★★☆	★★★★☆	★★☆☆☆
RASA	★★★★★	★★☆☆☆	★★☆☆☆	★★★★★

- Commercial tools are well-suited for simple but effective chatbots
- RASA framework allows more customization and personalized experience (while requiring more technical skills)

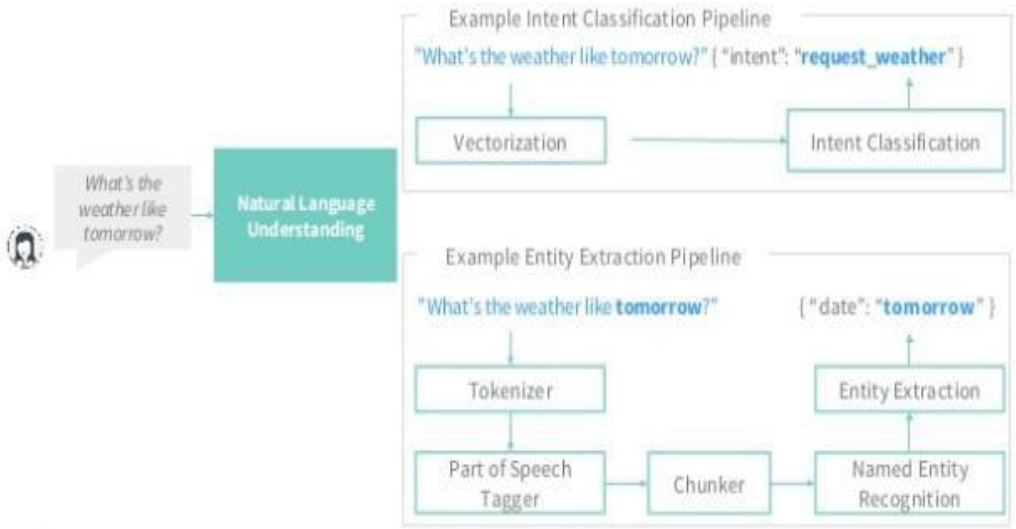
https://summalinguae.com/guides/building-chatbots/ (latest access: April 2021)

The RASA framework

- **Rasa NLU (Natural Language Understanding)**: interpret the input provided by the user in the form of structured data.
- Open-source natural language processing tool for intent classification (decides what the user is asking), extraction of the entity from the bot in the form of structured data and helps the chatbot understand what user is saying.

The RASA framework

- RASA NLU is the part that handles intent classification, entity extraction, and response retrieval

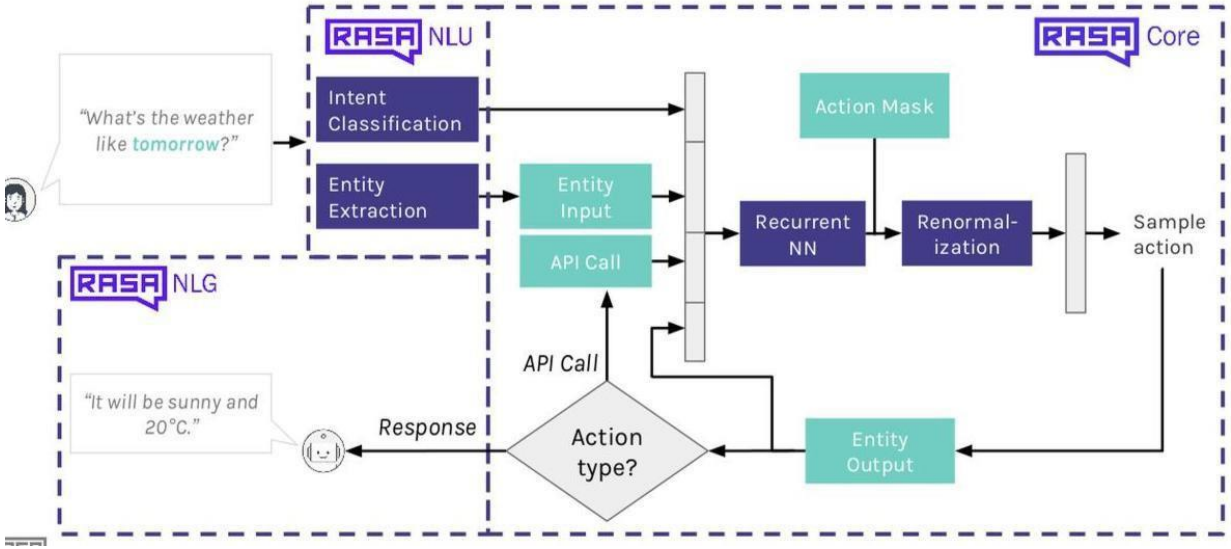


The RASA framework

- **Rasa Core**: decide the next set of actions performed by the chatbot.
- It is a framework with machine learning-based dialogue management.
- It takes the structured input from the NLU and predicts the next best action using a probabilistic model like LSTM neural network rather than if/else statement.
- Rasa Core and Rasa NLU are independent of each other and can be used separately.

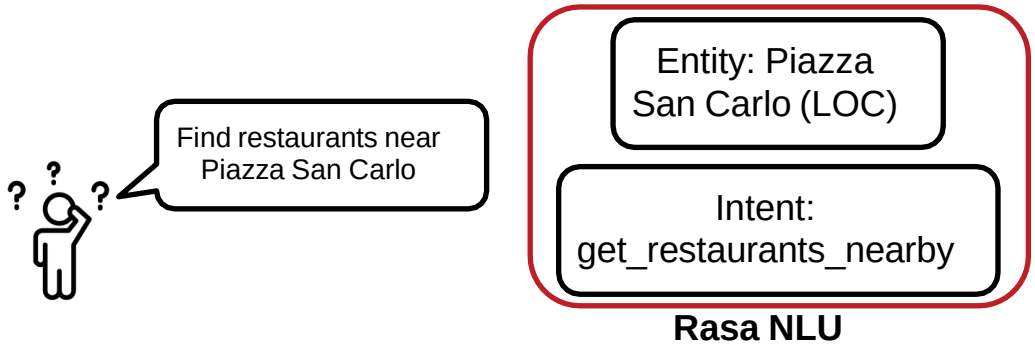
The RASA framework

- RASA Core entails the in-depth analysis of the user input
- RASA NLG exploits neural generative models to provide response in natural language



The RASA framework

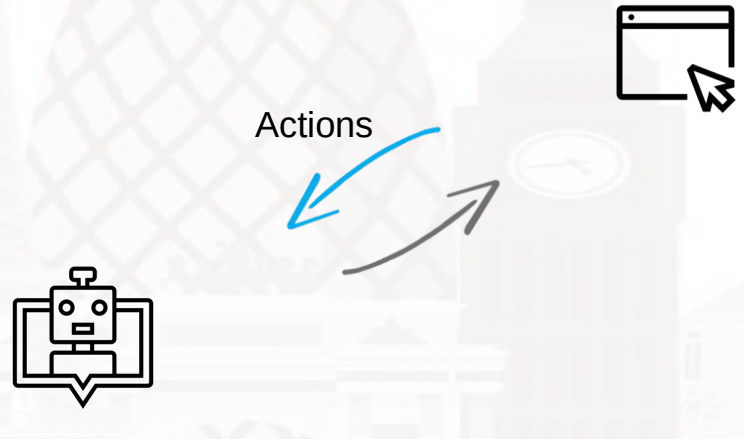
- The main goal of the NLU module is to extract structured data from raw text



The RASA framework

- Actions are tasks that can be accomplished by the bot
- They are often associated to a user intent

«Get the weather in Rome»
- Intent: **get_weather**
- Action: **look_for_weather_by_location**



Icons credit: Creative Mania and Clea Doltz from the Noun Project

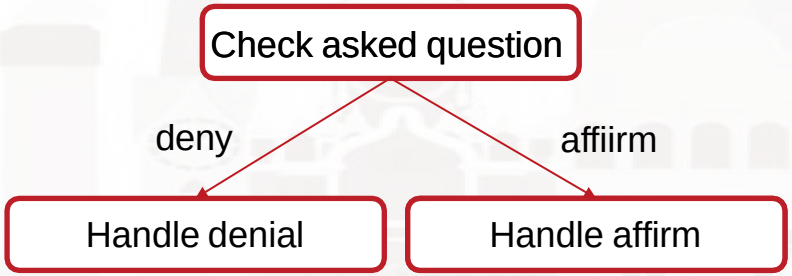
The RASA frameworkc

- User-bot interactions are defined by using **stories**
- A story is a sample **interaction between user and bots** in terms of intents and actions

```
stories:
-story: collect restaurant booking
info steps:
- intent: greet # user message with no entities
- action: utter_ask_howcanhelp
- intent: inform # user message with entities
entities:
- location: "rome"
- price: "cheap"
- action: utter_ask_cuisine
- intent: inform # expectations from the user input
entities:
- cuisine: "spanish"
- action: utter_ask_num_people
```

The RASA framework

- Interactions usually follow different flows
- **Checkpoints** can be used to separate actions




```
stories:
-story: beginning of flow steps:
- intent: greet
- action: action_ask_user_question
- checkpoint: check_asked_question
- story: handle user affirm
steps:
- checkpoint: check_asked_question
- intent: affirm
- action: action_handle_affirmation
- checkpoint: check_flow_finished
- story: handle user deny
steps:
- checkpoint: check_asked_question
- intent: deny
- action: action_handle_denial
- checkpoint: check_flow_finished
- story: finish flow
steps:
- checkpoint: check_flow_finished
- intent: goodbye
- action: utter_goodbye
```

The RASA framework

- Domains specify the context in which the bot operates
- Domain file specifies the **intents**, entities, slots, **responses**, forms, and actions your bot should know about

intents:
- affirm
- deny
- greet
- thankyou
- goodbye
- search_concerts
- search_venues
- compare_reviews
- bot_challenge

responses:

utter_greet:
-text: "Hey there!"
utter_goodbye:
-text: "Goodbye :("
utter_default:
-text: "Sorry, I didn't get that, can you rephrase?"
utter_youarewelcome:
-text: "You're very welcome."
utter_iamabot:
- text: "I am a bot, powered by Rasa."

The RASA domains

- Domains specify the context in which the bot operates
- Domain file specifies the intents, entities, **slots**, responses, forms, and **actions** your bot should know about

slots:

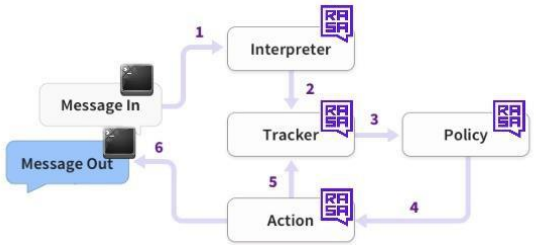
```
concerts:
    type: list
    influence_conversation: false
venues:
    type: list  influence_conversation:
    false
likes_music:
    type: bool
    influence_conversation: true
```

actions:

- action_search_concerts
- action_search_venues
- action_show_concert_reviews
- action_show_venue_reviews
- action_set_music_preference

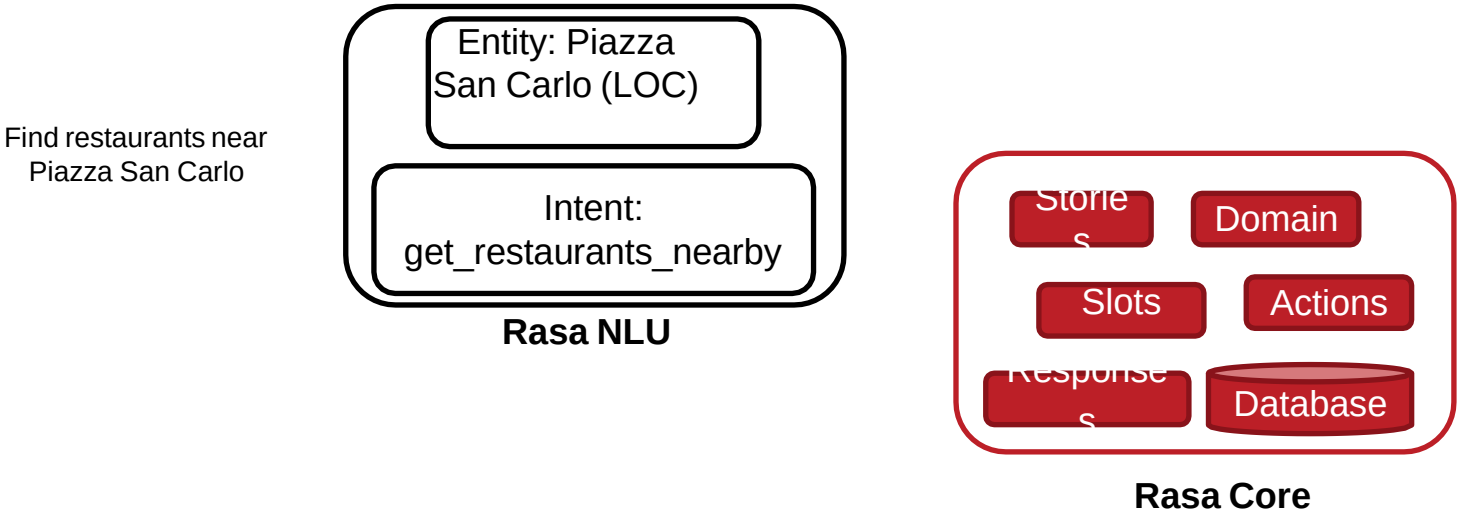
The RASA tracker

- The tracker stores the bot's memory
- Memory can be used to perform actions based on previous interactions
- Previous interactions contain relevant information about the next actions
- The tracker makes it possible to link interactions and let information flows



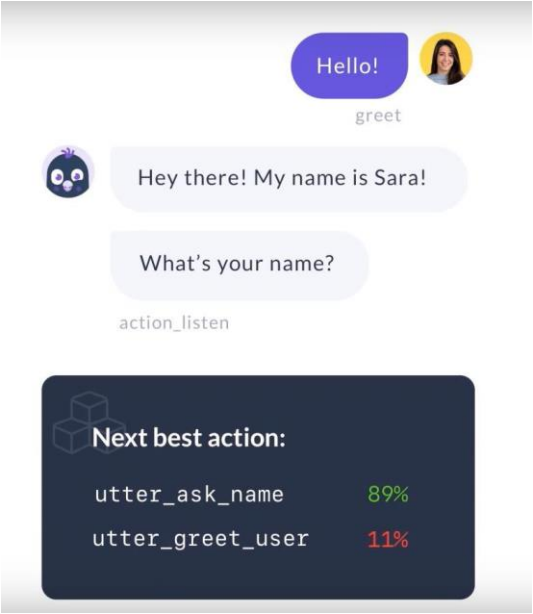
The RASA core

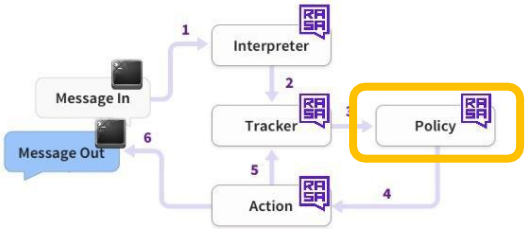
- The core module manages dialogues between user and chatbot
- It creates a probability model that decides the set of actions based on previous interactions



The RASA policies

- Policies are used to choose which action to take at each step
- Machine learning- and rule-based policies can be used simultaneously to compute the probability estimate



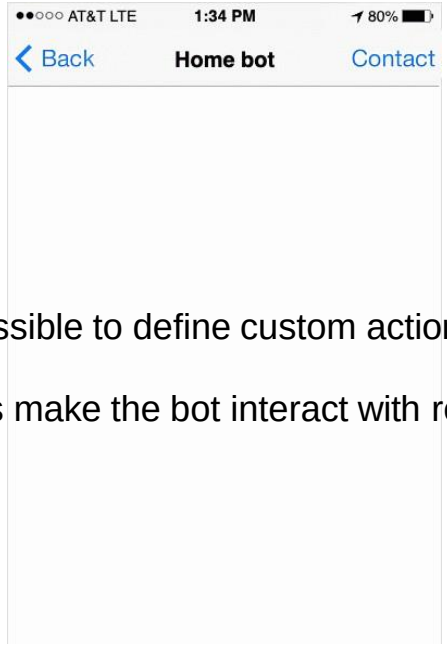


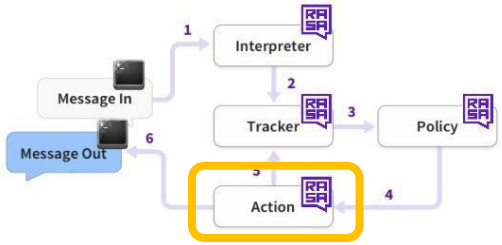
max_history:
parameter to control how much dialogue history the model need to consider

Form policy:
Can be used to let the bot fill existing forms requesting information to the user

The RASA actions

- After each user message, the model will predict an action that the assistant should perform next.

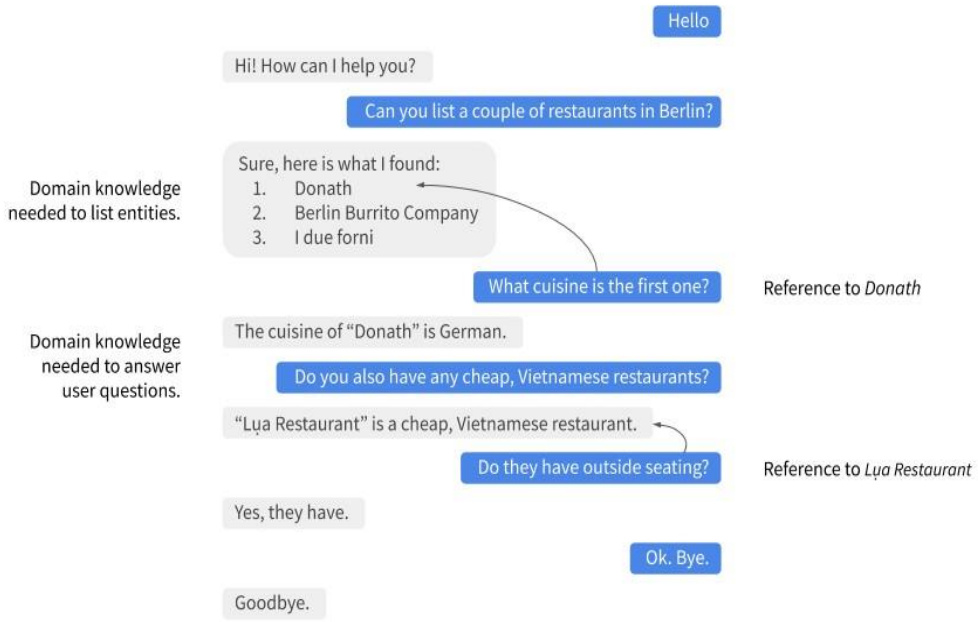


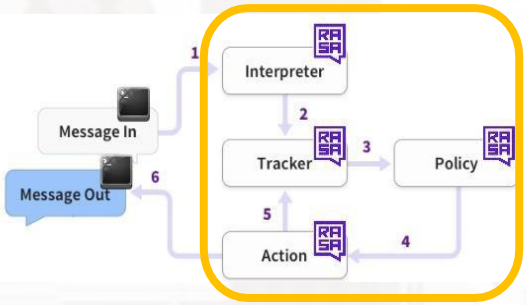


- It is possible to define custom actions to run specific portions of code
- Actions make the bot interact with real or digital environment

The RASA core pipeline example

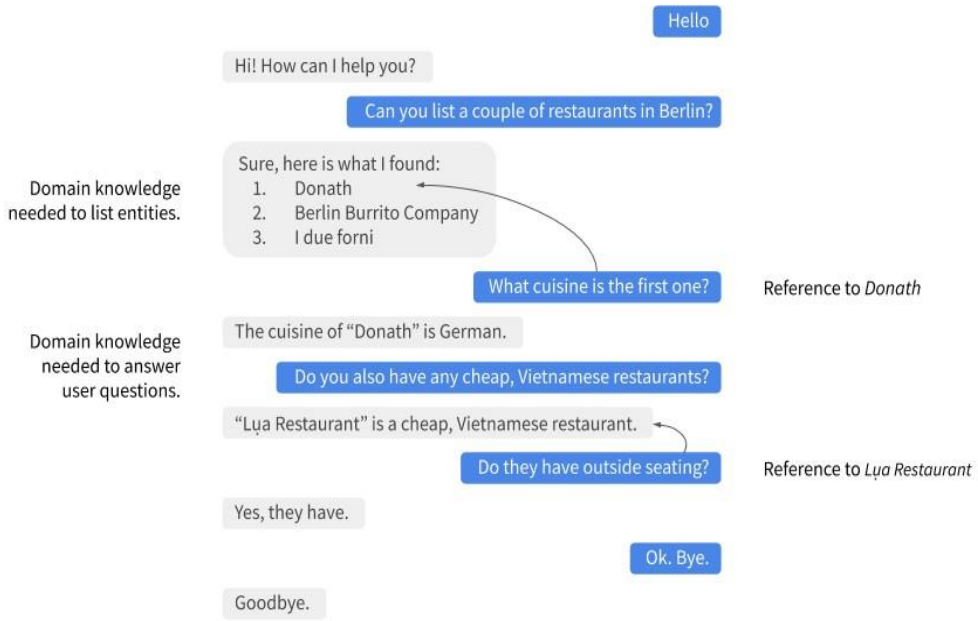
- After each user interaction user message, the core module predicts the action that should be performed next.

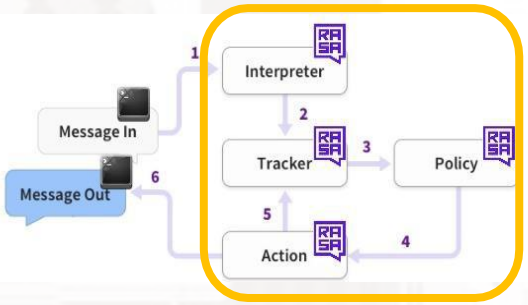




The RASA Natural Language Generation

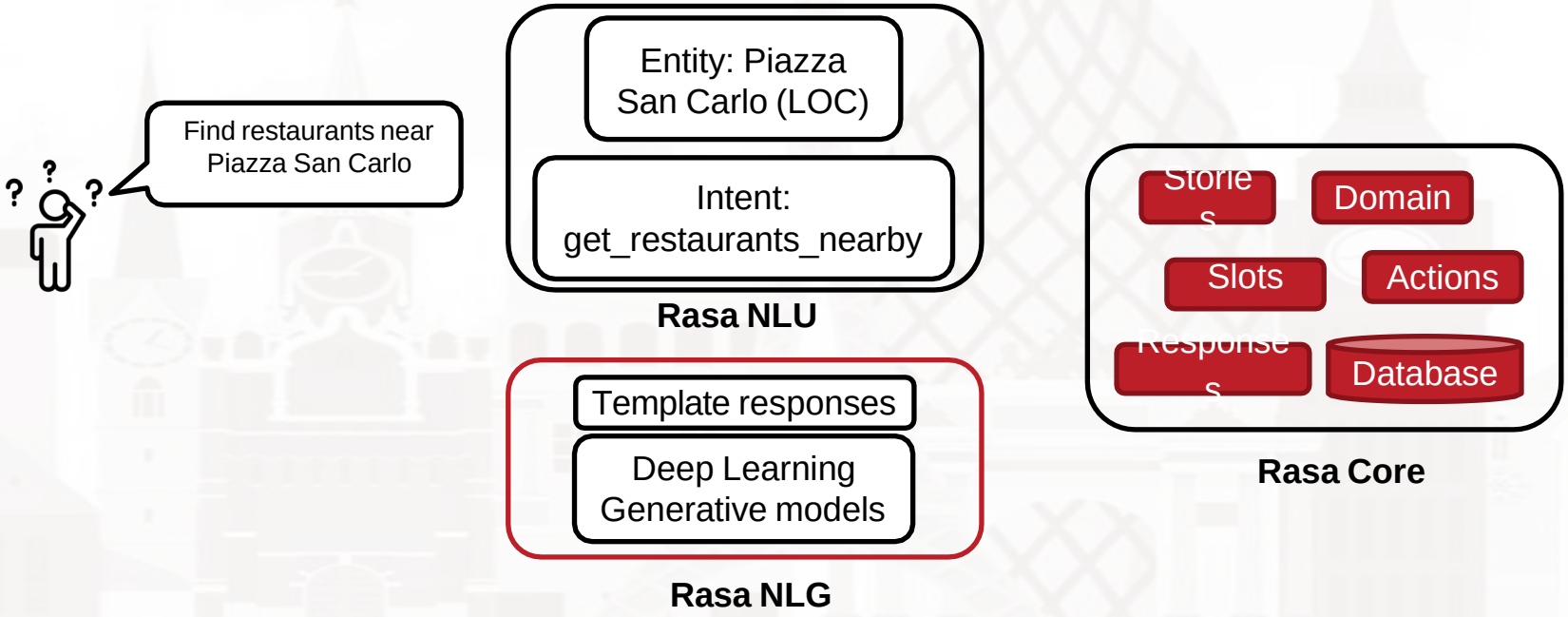
- After each user interaction user message, the core module predicts the action that should be performed next.





The RASA Natural Language Generation

- The generation module is the «mouth» of the chatbot
- It generates the responses that should be given to the user



Template-driven NLU

- Template-based models exploit variables to contextualize the responses
- Answers are mostly static and little context-dependent

```
templates:
    utter_greet:
    - "hey there {name}!" # variable filled by slot with the same name utter_goodbye:
      - "goodbye 😢"
    - "bye bye 😢" # multiple templates allow the bot to randomly pick one utter_default:
      - "default message"
```


Template-driven NLU

- The dynamic part of the template is limited
- Random variations should be defined a priori
- Interaction with the bot is not natural

Deep Learning-based NLU

- Deep learning architecture are able to generate responses in natural language
- Similar to human conversations, at each interaction answers may be different

Example from Facebook Blenderbot-3B
(https://huggingface.co/facebook/blenderbot-3B)

Additional reading on AI chatbot survey

- A Comparison of Natural Language Understanding Platforms for Chatbots in Software Engineering. CoRR abs/2012.02640 (2020)
- Please read the paper: https://arxiv.org/pdf/2012.02640.pdf

AI Chatbots – Ex. 1

An IT company provides online customer assistance using an AI chatbot. Known customers intents are collected in a csv dataset. Each intent is enriched with a single textual example. The chatbot relies on BERT text encoder.

Question:

Describe an unsupervised pipeline that includes the main chatbot steps.

Ex. 1 solution (a)

Given a customer request and the intent dataset:

1. Apply Speech-To-Text transformation to get the speech transcription of the request
2. Detect the source language and perform Machine Translation (if need be)
3. Get the textual example associated with each intent
4. Apply pretrained monolingual BERT encoding to each intent
   - Adopt the model fine-tuned for sentence similarity
5. Apply monolingual BERT encoding to the customer request
   - Adopt the model fine-tuned for sentence similarity
6. Compute pairwise request-intent similarities
7. If the maximum similarity level is above a minimum threshold:
   - Retrieve the most similar intent
8. Otherwise
   - Ask for request reformulation
9. If the intent requires an identifiable entity as parameter
   - Apply Named Entity Recognition
     - If no entity found
       - Request for the entity to the end-user

Ex. 1 solution (b)

Given a customer request

1. Apply Speech-To-Text transformation to get the speech transcription of the request
2. Truncate the speech transcription (if need be)
3. Get the textual descriptions of the intent
4. Encode each intent using Multilingual BERT (fine-tuned the model for sentence similarity)
   - E.g., Multilingual S-BERT (https://github.com/UKPLab/sentence-transformers)
5. Encode each customer request using Multilingual BERT
6. Compute pairwise request-intent similarities
7. If the maximum similarity level is above a minimum threshold:
   - Retrieve the most similar intent
8. Otherwise
   - Ask for request reformulation
9. If the intent requires an identifiable entity as parameter
   - Apply Named Entity Recognition
     - If no entity found
       - Request for the entity to the end-user

AI Chatbots – Ex. 2

An IT company provides online customer assistance using an AI chatbot. Known customers intents are collected in a csv dataset. Each intent is enriched with 500 textual examples. The chatbot relies on a state-of-the-art transformer-based model.

Question:

Describe a supervised pipeline that includes the main chatbot steps.

Ex. 2 solution

Given a customer request and the intent dataset:

1. Apply Speech-To-Text transformation to get the speech transcription of the request
2. Detect the source language and perform Machine Translation (if need be)
3. Get the textual example associated with each intent
4. Fine-tune the RoBERTa model using the annotated data
   - Use the sequence classification version
5. Apply the fine-tuned RoBERTa model
   - Inference on user request
6. If the returned class is *Not Found*:
   - Ask for request reformulation
7. If the intent requires an identifiable entity as parameter
   - Apply Named Entity Recognition
     - If no entity found
       - Request for the entity to the end-user

Acknowlegdements and copyright license

- Copyright licence
  - Attribution + Noncommercial + NoDerivatives

- Acknowledgements
  - I would like to thank Dr. Moreno La Quatra, who collaborated to the writing and revision of the teaching content
- Affiliation
  - The author and his staff are currently members of the Database and Data Mining Group at Dipartimento di Automatica e Informatica (Politecnico di Torino) and of the SmartData interdepartmental centre
    - https://dbdmg.polito.it
    - https://smartdata.polito.it

Thank you!

