Text Classification and Intent Recognition

[DAT640] Information Retrieval and Text Mining

Petra Galuscakova
University of Stavanger

October 22, 2024

In this module

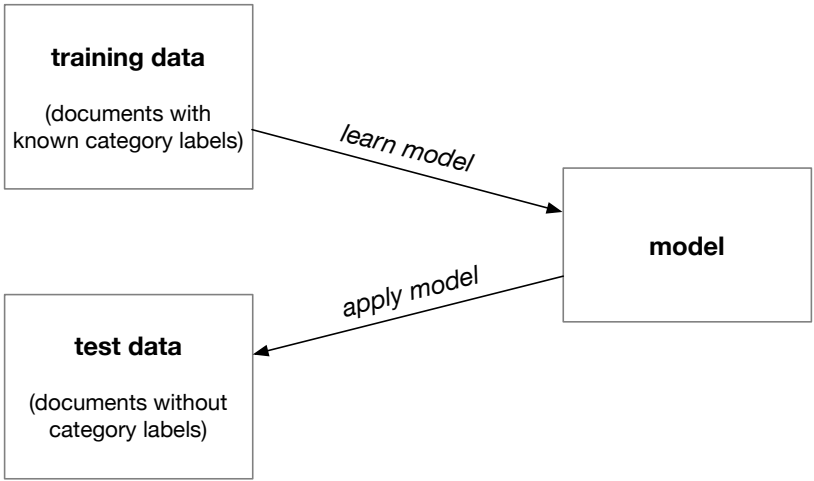
1. Text classification
2. Statistical Approaches
3. Neural Approaches
4. Applications to Natural Language Understanding (NLU)
5. Tools and Applications for Intent Recognition

Text classification

Text classification

- **Classification** is the problem of assigning objects to one of several predefined categories
  - One of the fundamental problems in machine learning
  - It is performed using the training dataset (instances with known category membership)
- In **text classification** (or **text categorization**) the objects are text documents or strings
- Binary classification (two classes, 0/1 or -/+)
  - E.g., deciding whether an email is spam or not
- Multiclass classification (n classes)
  - E.g., Categorizing news stories into topics (finance, weather, politics, sports, etc.)

General approach



Formally

- Given a training sample $(\boldsymbol{X}, y)$, where $\boldsymbol{X}$ is a set of documents with corresponding labels y, from a set $\boldsymbol{Y}$ of possible labels, the task is to learn a function $f(\cdot)$ that can predict the class $y' = f(x)$ for an unseen document x.

Families of approaches

- Feature-based approaches (“traditional” or “statistical” machine learning)
- Neural approaches (“deep learning”)

Statistical Approaches

Question

What could be used as features in text classification?

Features for text classification

- Use words as features (**bag-of-words**)
  - Words will be referred to as **terms**
- Values can be, e.g., binary (term presence/absence) or integers (term counts)
- Documents are represented by their **term vector**
- **Document-term matrix** is huge, but most of the values are zeros; stored as a sparse matrix

	t_1	t_2	t_3	$\ldots$	t_m
d_1	1	0	2		0
d_2	0	1	0		2
d_3	0	0	1		0
$\ldots$					
d_n	0	1	0		0

Document-term matrix

Bayes Rule

- Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred.
- $P(A|B) = \frac{P(B|A)*P(A)}{P(B)}$ where A and B are events and $P(B)\neq 0$
- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as evidence.
- P(A) is the priori of A (the prior probability, i.e. Probability of event before evidence is seen).
- P(B) is Marginal Probability: Probability of Evidence.
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.
- P(B|A) is Likelihood probability i.e the likelihood that a hypothesis will come true based on the evidence.

Naive Bayes

- Estimating the probability of document $\boldsymbol{x}$ belonging to class y

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$$

- $P(\boldsymbol{x}|y)$ is the class-conditional probability
- $P(y)$ is the prior probability
- $P(\boldsymbol{x})$ is the evidence (note: it's the same for all classes)

Naive Bayes classifier

- Estimating the class-conditional probability $P(y|\boldsymbol{x})$
  - $\boldsymbol{x}$ is a vector of term frequencies $\{x_1, \ldots, x_n\}$

$$P(\boldsymbol{x}|y) = P(x_1, \ldots, x_n|y)$$

- "Naive" assumption: features (terms) are independent:

$$P(\boldsymbol{x}|y) = \prod_{i=1}^{n} P(x_i|y)$$

- Putting our choices together, the probability that $\boldsymbol{x}$ belongs to class y is estimated using:

$$P(y|\boldsymbol{x}) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Estimating prior class probabilities

- $P(y)$ is the probability of each class label
- It is essential when class labels are imbalanced

Estimating feature distribution

- How to estimate $P(x_i|y)$?
- Maximum likelihood estimation: count the number of times a term occurs in a class divided by its total number of occurrences

$$P(x_i|y) = \frac{c_{i,y}}{c_i}$$

  - $c_{i,y}$ is the number of times term x_i appears in class y
  - c_i is the total number of times term x_i appears in the collection
- But what happens if $c_{i,y}$ is zero?!

Smoothing

- Ensure that $P(x_i|y)$ is never zero
- Simplest solution: Laplace ("add one") smoothing

$$P(x_i|y) = \frac{c_{i,y} + 1}{c_i + m}$$

  - m is the number of classes (i.e. the number of unique words in the vocabulary)

Practical considerations

- In practice, probabilities are small, and multiplying them may result in numerical underflows
- Instead, we perform the computations in the log domain

$$\log P(y|\boldsymbol{x}) \propto \log P(y) + \sum_{i=1}^{n} \log P(x_i|y)$$

Naive Bayes Example

Whether	Play
Sunny	No
Sunny	No
Overcast	Yes
Rainy	Yes
Rainy	Yes
Rainy	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table

Whether	No	Yes
Overcast		4
Sunny	2	3
Rainy	3	2
Total	5	9

Likelihood Table 1

Whether	No	Yes		
Overcast		4	=4/14	0.29
Sunny	2	3	=5/14	0.36
Rainy	3	2	=5/14	0.36
Total	5	9		
	=5/14	=9/14		
	0.36	0.64		

Likelihood Table 2

Whether	No	Yes	Posterior Probability for No	Posterior Probability for Yes
Overcast		4	0/5=0	4/9=0.44
Sunny	2	3	2/5=0.4	3/9=0.33
Rainy	3	2	3/5=0.6	2/9=0.22
Total	5	9		

Naive Bayes Example cont.

- Now suppose you want to calculate the probability of playing when the weather is overcast.
- Probability of playing:
  - $P(Yes|Overcast) = \frac{P(Overcast|Yes)P(Yes)}{P(Overcast)}$
- Calculate Prior Probabilities:
  - P(Overcast) = 4/14 = 0.29
  - P(Yes)= 9/14 = 0.64
- Calculate Posterior Probabilities:
  - P(Overcast | Yes) = 4/9 = 0.44
- P (Yes | Overcast) = 0.44 * 0.64 / 0.29 = 0.98

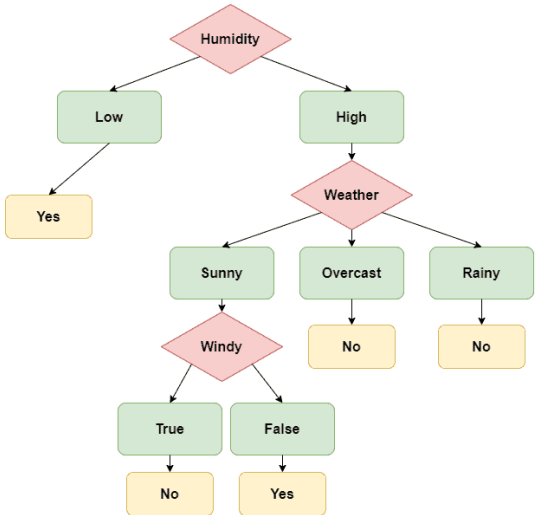
Exercise

E13-1 Naive Bayes Exercise

Decision Tree

- A decision tree is formed by a collection of value checks on each feature. During inference, we check each individual feature and follow the branch that corresponds to its value. This traversal continues until a terminal node is reached, which contains a decision.
- Use the features in descending order of importance and even discard 'useless' features that don't increase accuracy.
- A commonly used metric to define the importance of a feature is the informational gain (typically it is calculated by an entropy)
- For a set of possible feature values $y_1, ..., y_k$ with associated probabilities $p(y_1), ..., p(y_k)$, the information we get is:
$I(p(y_1), ..., p(y_k)) = -\sum_{i=1}^{k} p(y_i) log_2 p(y_i)$

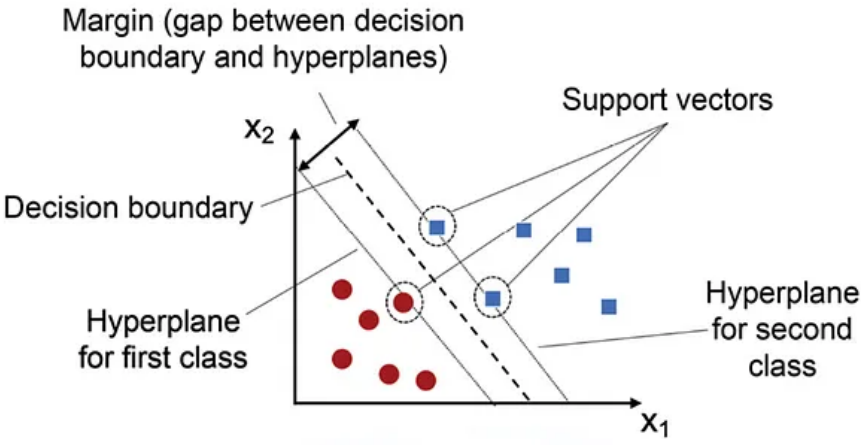
Decision Tree



Decision Trees Example

- When a feature's values divide the data entries into approximately equal groups, then the informational content is higher, and the feature should be used early in the tree.
- This approach also has the benefit of creating relatively balanced decision trees. As a result, paths from the root to the terminal nodes are shorter, and therefore inference is faster.
- However, unless properly pruned, decision trees tend to overfit the training data.
- Decision trees are good for visualizing the decision.

Support Vector Machine



Support Vector Machine

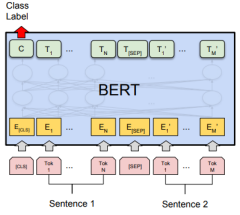
- SVMs work by finding the optimal hyperplane that separates data points into different classes.
- A hyperplane is a decision boundary that separates data points into different classes in a high-dimensional space. In two-dimensional space, a hyperplane is simply a line that separates the data points into two classes. It can be used to make predictions on new data points by evaluating which side of the hyperplane they fall on.
- Data points on one side of the hyperplane are classified as belonging to one class, while data points on the other side of the hyperplane are classified as belonging to another class.

Support Vector Machine

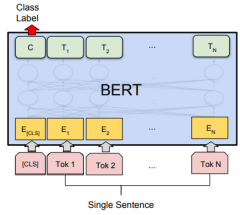
- A margin is the distance between the decision boundary (hyperplane) and the closest data points from each class.
- The goal of SVMs is to maximize this margin while minimizing classification errors.
- A larger margin indicates a greater degree of confidence in the classification, as it means that there is a larger gap between the decision boundary and the closest data points from each class.

Neural Approaches

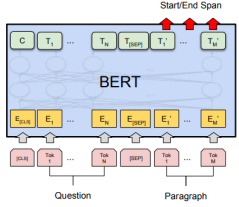
BERT Classification recap.



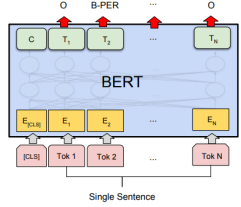
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA

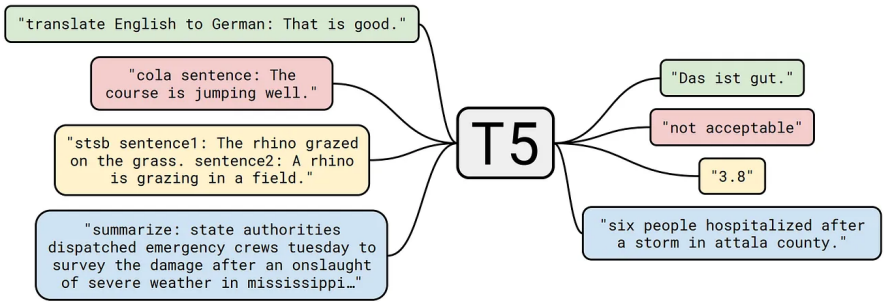


(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

T5 Classification recap.



T5 Classification

- Trained using a prompt learning.
- An encoder-decoder model is fine-tuned with a downstream task taking as input this input configuration, and generating an output sequence whose last token is equal to *True* or *False*, depending on whether belonging to the class.
- The classification probability is computed by normalising only the *False* and *True* output probabilities, computed over the whole vocabulary, with a *softmax* operation

Generative Models Classification recap.

- Zero shot example: Create a mail classifier with the categories SPAM, WORK and PERSONAL. Classify following message: I hope you are having a great week! Just writing to check in on this. Do not write code, only provide the category on the output.
- One shot example: Create a mail classifier with the categories SPAM, WORK and PERSONAL. Classify following message: I hope you are having a great week! Just writing to check in on this. Example of WORK: You're signed up to receive a daily report of some notifications from your Canvas account. Below is the report for Oct 14 : Do not write code, only provide the category on the output.

Applications to Natural Language Understanding (NLU)

Intent Recognition

- An intent in a task-oriented dialogue system is a label that captures the meaning of an utterance through the intention the user expresses in said utterance.
- Intents can be represented at different levels of granularity.
- The two main approaches for intent annotation are dialogue acts and domain-specific intents.
- The domain-specific intents are usually very fine-grained, with one intent corresponding to one task an agent can perform (e.g., BookFlight or GetWeather).

Intention and Dialogue Acts

- A dialogue act in task-oriented dialogue captures the general intention or speech act behind an utterance independent of domain or dialogue system.
- Common dialogue acts include inform, request, confirm, deny, request alts, ...
- Some dialogue acts include slot-value pairs that provide information about entities in the utterance.
- Example: inform(food="Italian") which could belong to an utterance where a user is looking for Italian restaurants.

Dialogue Acts Schema

Universal DA Schema	DSTC 2	M2M
inform	inform(x=y)	inform(x=y)
request	request(x)	request(x)
user-negate	negate(x=y)	negate(x=y)
sys-negate	negate(x=y)	negate(x=y)
user-hi	hello()	
user-hi + inform		greeting(x=y)
sys-hi	welcomemsg()	
reqalts	reqalts(), reqmore()	request_alts()
repeat	repeat()	cant_understand()
restart	restart()	
affirm	affirm()	affirm()
affirm + inform		affirm(x=y)
sys-impl-confirm	impl-conf(x=y)	
sys-expl-confirm	expl-conf(x=y), confirm-domain(x=y)	confirm(x=y)
user-confirm	confirm(x=y)	
sys-notify-failure	canthelp(), canthelp.exception()	notfiy_failure()
sys-notify-success		notify_success()
sys-offer	offer(x=y), select(x=y1,y2)	offer(x=y), select(x=y1,y2)
thank_you	thankyou()	thank_you()
bye	bye()	
ack	ack()	
deny	deny(x=y)	

Frame-Based Dialogue System

- A task-based dialogue system has the goal of helping a user solve a specific task.
- Task-based dialogue systems are based around frames.
- Frames are knowledge structures representing the details of the user's task specification.
- Each frame consists of a collection of slots, each of which can take a set of possible values.

Frame-Based Dialogue System

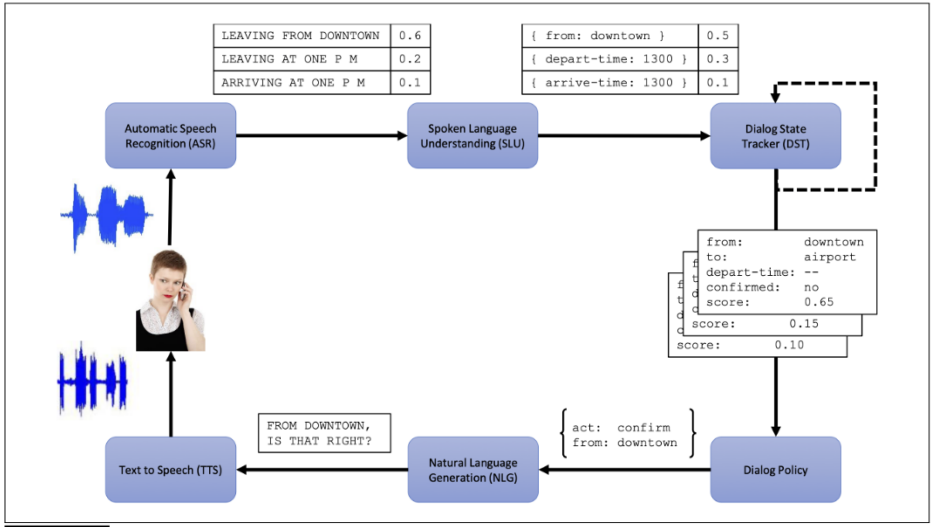


Figure 15.3 Architecture of a dialogue-state system for task-oriented dialogue from Williams et al. (2016).

Frame-Based Dialogue System

- The frame and its slots in a task-based dialogue system specify what the system needs to know to perform its task.
  - A hotel reservation system needs dates and locations.
  - An alarm clock system needs a time.
- The system's goal is to fill the slots in the frame with the fillers the user intends, and then perform the relevant action for the user (answering a question, or booking a flight).

Frame-Based Dialogue System

Slot	Type	Example Question
ORIGIN CITY	city	"From what city are you leaving?"
DESTINATION CITY	city	"Where are you going?"
DEPARTURE TIME	time	"When would you like to leave?"
DEPARTURE DATE	date	"What day would you like to leave?"
ARRIVAL TIME	time	"When do you want to arrive?"
ARRIVAL DATE	date	"What day would you like to arrive?"

Figure 15.4 A frame in a frame-based dialogue system, showing the type of each slot and a sample question used to fill the slot.

Disambiguation

- Many domains require multiple frames.
- Besides frames for car or hotel reservations, we might need other frames for things like general route information (for questions like Which airlines fly from Boston to San Francisco?),
- The system must be able to disambiguate which slot of which frame a given input is supposed to fill.

Intent Determination

- Domain Classification: is this user for example talking about airlines, programming an alarm clock, or dealing with their calendar?
- Intent Determination: what general task or goal is the intent determination user trying to accomplish? For example the task could be to Find a Movie, or Show a Flight, or Remove a Calendar Appointment.
- Together, the domain classification and intent determination tasks decide which frame we are filling.
- Slot Filling: extract the particular slots and fillers that the user intends system to understand from their utterance with respect to their intent.

Intent Determination Example

- Show me morning flights from Boston to San Francisco on Tuesday.
- DOMAIN: AIR-TRAVEL
- INTENT: SHOW-FLIGHTS
- ORIGIN-CITY: Boston
- DEST-CITY: San Francisco
- ORIGIN-DATE: Tuesday
- ORIGIN-TIME: morning

Handwritten Rules for Slot-filling

- The simplest dialogue systems use handwritten rules for slot-filling
- wake me (up) | set (the|an) alarm | get me up

Supervised Machine-learning for Slot-filling

- Each sentence in a training set is annotated with slots, domain, and intent, and a sequence model maps from input words to slot fillers, domain and intent.
- For example we'll have pairs of sentences that are labeled for domain (AIRLINE) and intent (SHOWFLIGHT), and are also labeled with BIO representations for the slots and fillers.
- beginning (B) and inside (I) of each slot label, and one for tokens outside (O) any slot label

```
O O    O  O   O  B-DES I-DES     O  B-DEPTIME I-DEPTIME  O        AIRLINE-SHOWFLIGHT
I want to fly to San   Francisco on Monday    afternoon  please   EOS
```

BERT for Slot-filling

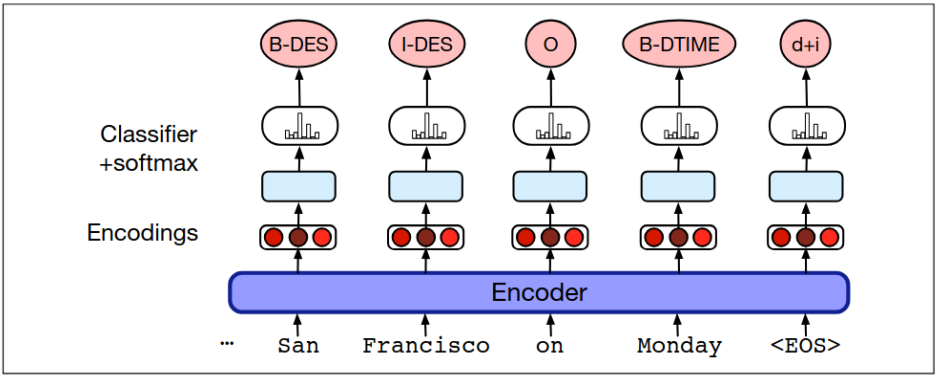


Figure 15.5 Slot filling by passing input words through an encoder, and then using a linear or feedforward layer followed by a softmax to generate a series of BIO tags. Here we also show a final state: a domain concatenated with an intent.

BERT for Slot-filling

- The input words $w_1 ... w_n$ are passed through a pretrained language model encoder, followed by a feedforward layer and a softmax at each token position over possible BIO tags, with the output a series of BIO tags $s_1 ... s_n$.
- We generally combine the domain-classification and intent-extraction tasks with slot-filling by adding a domain concatenated with an intent as the desired output for the final EOS token.
- We can make a very simple frame-based dialogue system by wrapping a small amount of code around this slot extractor. Mainly we just need to ask the user questions until all the slots are full, do a database query, then report back to the user, using hand-built templates for generating sentences.

Tools and Applications for Intent Recognition

Domain-specific Intent Datasets

- ATIS (Airline Travel Information Systems): a dataset with recordings of flight reservations that has 21 intent labels in the flight domain.
- SNIPS (Natural Language Understanding benchmark): a dataset labeled with seven intents related to a virtual assistant.
- HWU64: a dataset with 64 virtual assistant intents, natural language data for human-robot interaction.
- CLINC150: a dataset with 150 intents spanning 10 domains like travel, dining, and small talk, focused on evaluating the performance of intent classification systems in the presence of "out-of-scope" queries,
- Facebook's multilingual dataset: utterances in the weather, alarm, and reminder domains with 12 intents.

Intent Recognition Tools

Table 3.1: Features related to intent classification in NLU platforms. *Full CA* indicates whether the platform provides a full conversational agent or just a NLU service.

Platform	Full CA	Multi-Intent Support	Patterns/ RegEx	Pre-Built Intents	Cloud-Based	Configurability
Wit.ai		✓		✓	✓	
Dialogflow	✓			✓	✓	
LUIS		✓	✓	✓	✓	
Lex	✓	✓		✓	✓	
Watson	✓	✓		✓	✓	
RASA	✓	✓	✓			✓

Intent Recognition in Rasa

- DialogueKit is based on Rasa
- Intent Recognition: https://rasa.com/docs/rasa/components/#intent-classifiers
- Entity Recognition: https://rasa.com/docs/rasa/components/#entity-extractors
- Also supports training: https://medium.com/mantisnlp/implementing-a-custom-intent-classification-model-with-rasa-5ab4283b5b14 and https://www.restack.io/p/intent-recognition-knowledge-rasa-cat-ai

BERT Models for Intent Classification

- Hugging face: https://huggingface.co/models?other=intent-classification
- TOD-BERT: Pre-trained Natural Language Understanding for Task-Oriented Dialogue
  - For pre-training they collect 9 multi-turn, human-human, task-oriented dialogue datasets with over 100,000 dialogues spanning 60 domains.
  - The authors fine-tune for 4 downstream tasks, including intent recognition and dialogue act prediction.
- ConvBERT: a model produced by further pre-training of BERT on 700 million conversations from online forums.

Summary

- Classification approaches (statistical and neural)
- Frame-based dialogue systems
- Intent classification

Reading and References

- *Speech and Language Processing, Chapter 4: Naive Bayes, Text Classification, and Sentiment*, Daniel Jurafsky and James H. Martin[1]
- *Speech and Language Processing, Chapter 15: Chatbots & Dialogue Systems*, Daniel Jurafsky and James H. Martin[2]

[1] `https://web.stanford.edu/~jurafsky/slp3/4.pdf`
[2] `https://web.stanford.edu/~jurafsky/slp3/15.pdf`