Rule-Based NLP

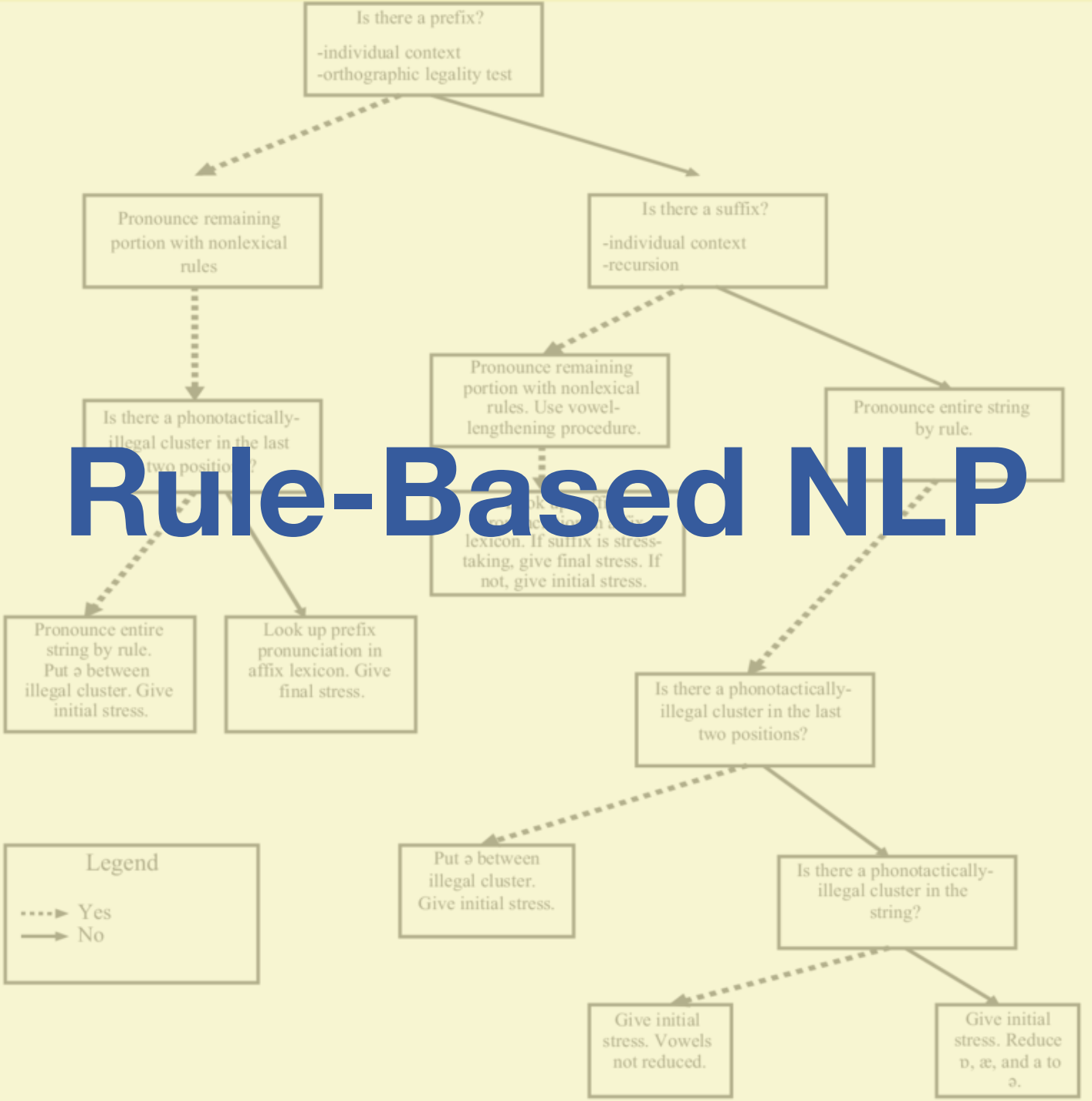
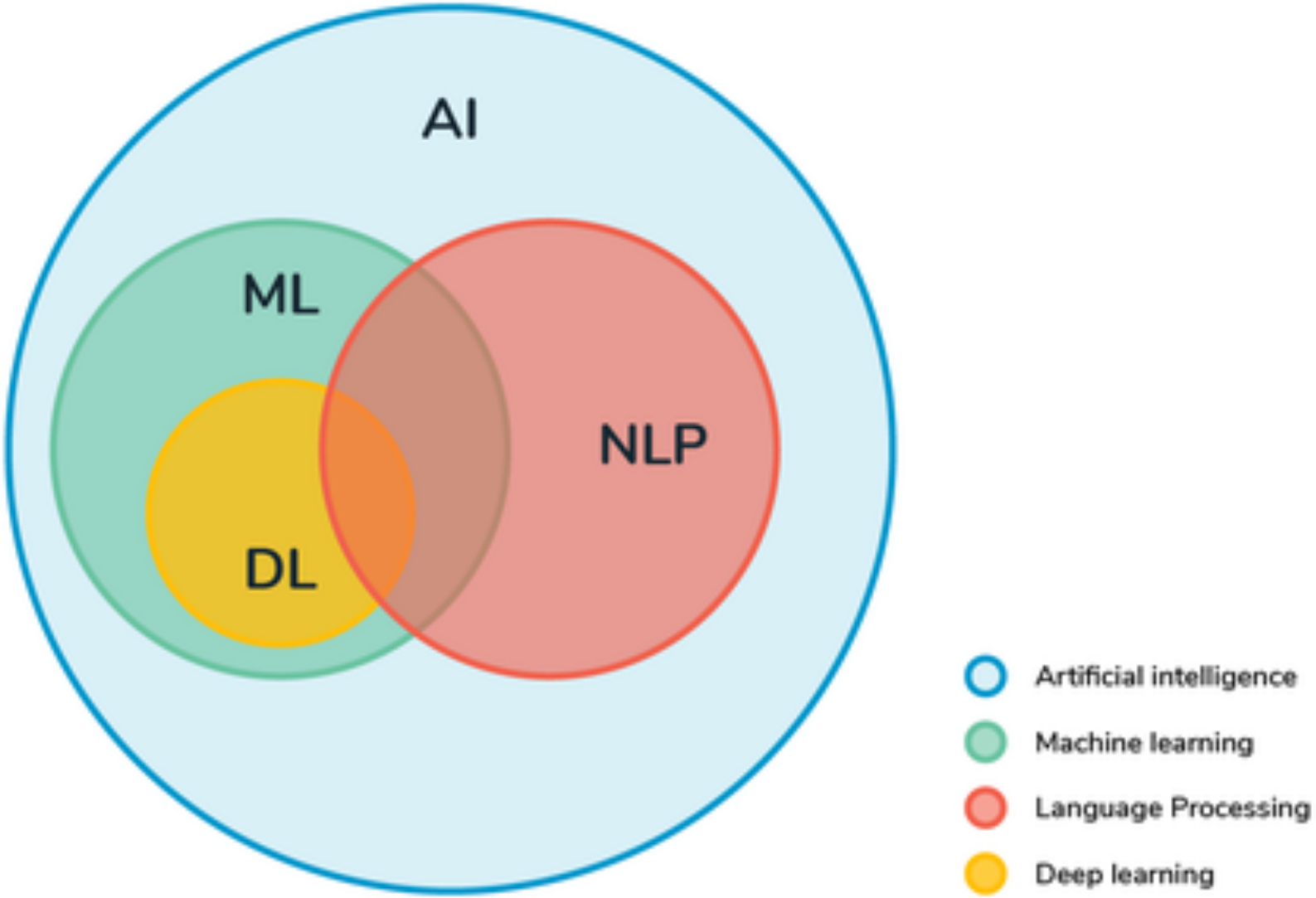


TABLE OF CONTENTS

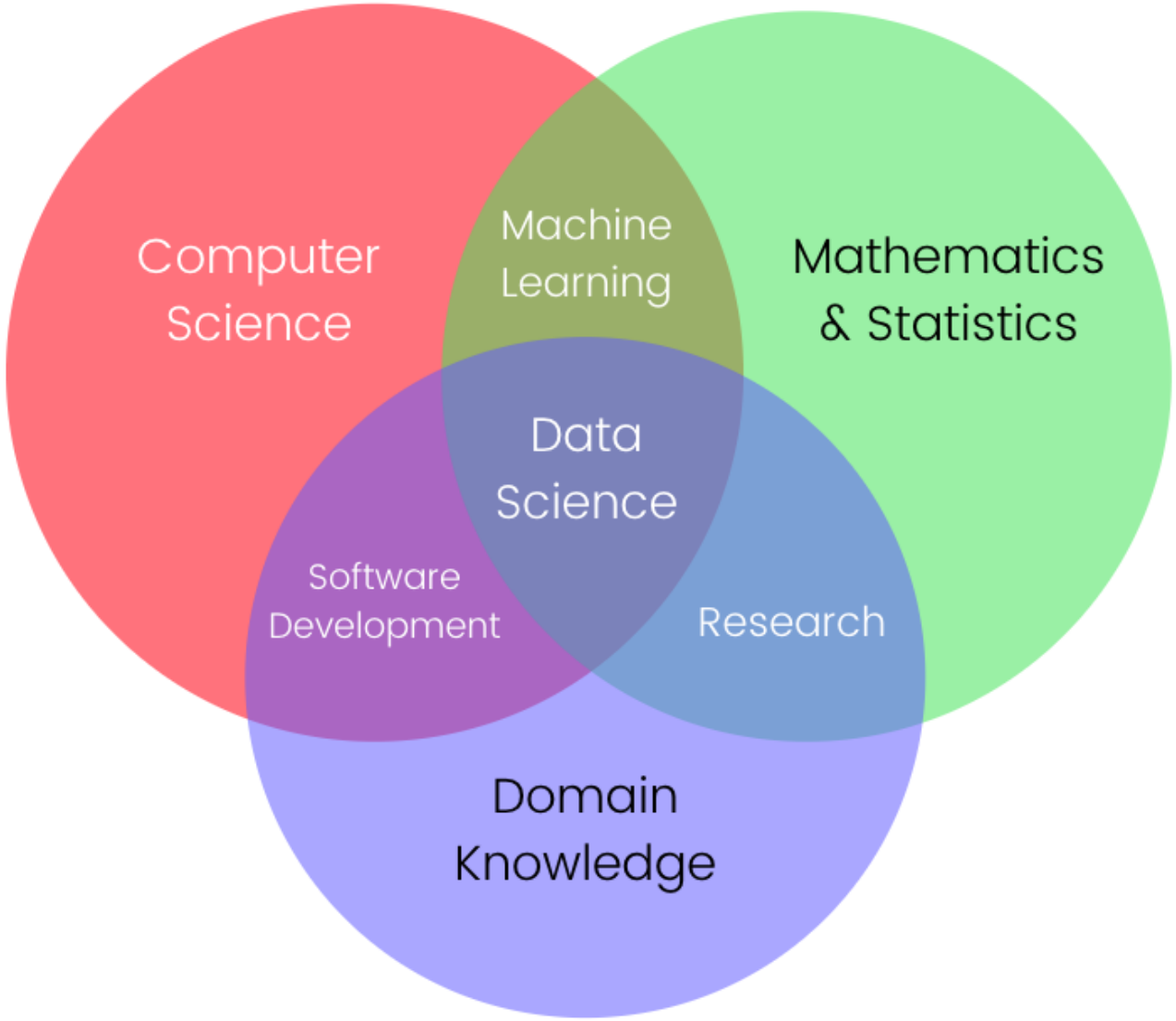
WHAT IS NLP ANYWAY?

By Method



WHAT IS NLP ANYWAY?

By Knowledge/
Skill Applied



COMPUTATIONAL LINGUISTICS VS NLP

Computational Linguistics — 80s-00s, dynamic programming, statistics, research-focused

NLP — 2010s +, machine learning, engineering and product-focused

HOW TO APPROACH AN NLP TASK

- Business problem (saves resources/generates income)

    «We want to analyze financial articles to predict stock prices»

HOW TO APPROACH AN NLP TASK

- Business problem (saves resources/generates income)

  «We want to analyze financial articles to predict stock prices»

- Linguistic problem (what role is language playing?)

  Gauge from certain phrases in text whether the stock price of a certain company will go up or down: extract relevant phrases, connect them to a particular company, decide if they are positive or negative

HOW TO APPROACH AN NLP TASK

- Business problem (saves resources/generates income)

  «We want to analyze financial articles to predict stock prices»

- Linguistic problem (what role is language playing?)

  Gauge from certain phrases in text whether the stock price of a certain company will go up or down: extract relevant phrases, connect them to a particular company, decide if they are positive or negative

- NLP task(s)

  - phrase extraction/text mining

  - Named Entity Recognition, co-reference

  - text classification

HOW TO APPROACH AN NLP TASK

- Business problem (saves resources/generates income)

  «We want to analyze financial articles to predict stock prices»

- Linguistic problem (what role is language playing?)

  Gauge from certain phrases in text whether the stock price of a certain company will go up or down: extract relevant phrases, connect them to a particular company, decide if they are positive or negative

- NLP task(s)

  - phrase extraction/text mining
  - Named Entity Recognition, co-reference
  - text classification

- NLP method (decide based on what and how much data you have)

  - rule-based phrase extraction
  - pre-trained neural network for NER, probabilistic classifier for co-reference
  - pre-trained sentiment classifier

HOW TO CHOOSE THE METHOD?

- available pre-trained models for a particular task
- amount of data available
- specificity of the task (task is too vague to have any associated data sets)
- connectedness of data to its labels (patterns in data)

class 1: I like cats.
class 2: I like dogs.

CONNECTEDNESS OF DATA TO ITS LABELS
EXAMPLE: MERCHANDISE

The variability of data is too high to be easily learned by ML. But adding a concreteness score to each noun may help.

COVERING SHIPMENT OF 15 MT SODIUM HEXAMETAPHOSPHATE
AT THE RATE OF USD 1000 PER MT. TOTALY VALUED USD 15000 AS PER SALES CONTRACT NO. EGHR3462HHR DATED 5-SEP-2020 (INCOTERMS 2010 CIF, HAMBURG PORT, GERMANY)

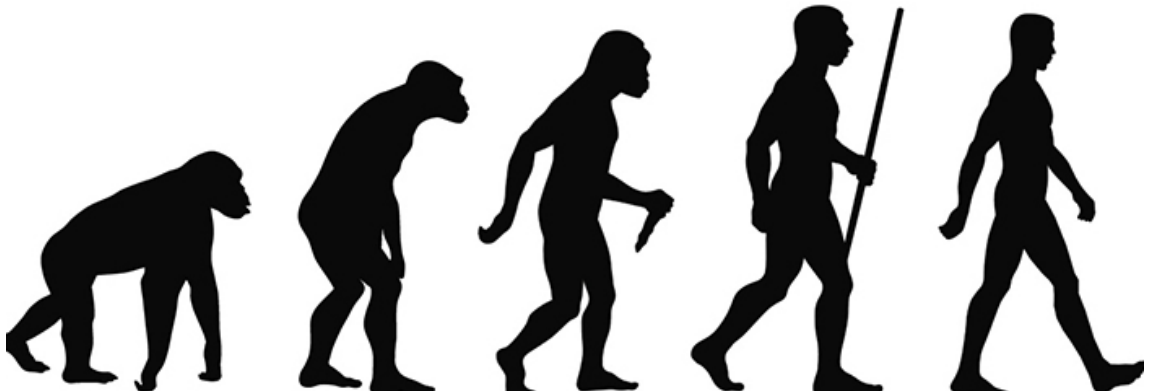
60,000 BOTTLES OF RED WINE
UNIT PRICE: USD4.00
CIF TORONTO/ORIGIN:FRENCH
PACKING: 40 CRATES WITH 60,000 BOTTLES

TERMS OF PRICE : FOB
ORIGIN : GERMANY
VOLKSWAGEN D24TIC ENGINE WITH ACCESSORIES

TERMS OF PRICE : FCA USA AIRPORT
COUNTRY OF ORIGIN : UNITED STATES AMERICA
NIGHT VISION BINOCULARS
DETAILS ARE AS PER PURCHASE ORDER NO : NFP123455

NLP ALGORITHM EVOLUTION

- Dictionary-based
- Rule-based
- Statistical
- Machine Learning
- Neural Networks

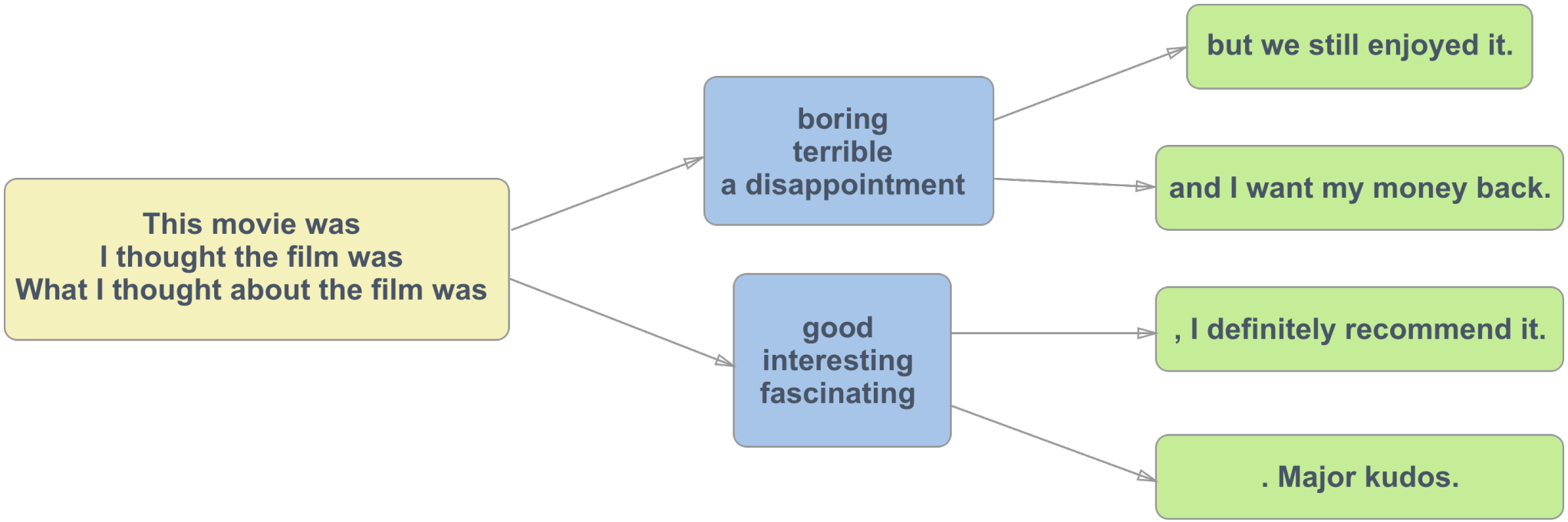


HEURISTIC APPROACH TO NLP PROBLEM SOLVING

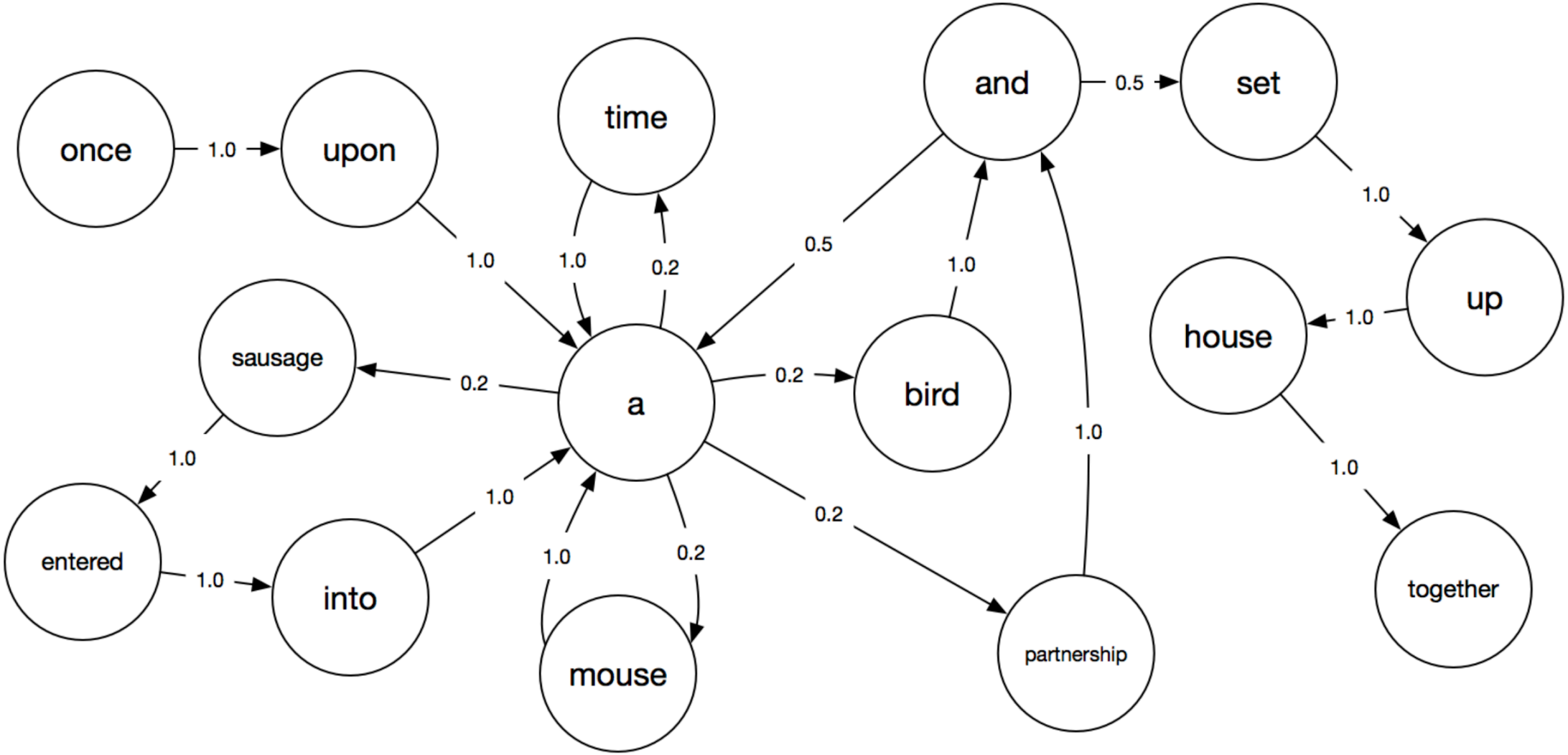
Rule-based: practical method not guaranteed to be optimal or perfect, but sufficient for reaching an immediate goal.

Statistical or ML: the decision making process is objective and is based on direct evidence from the given data.

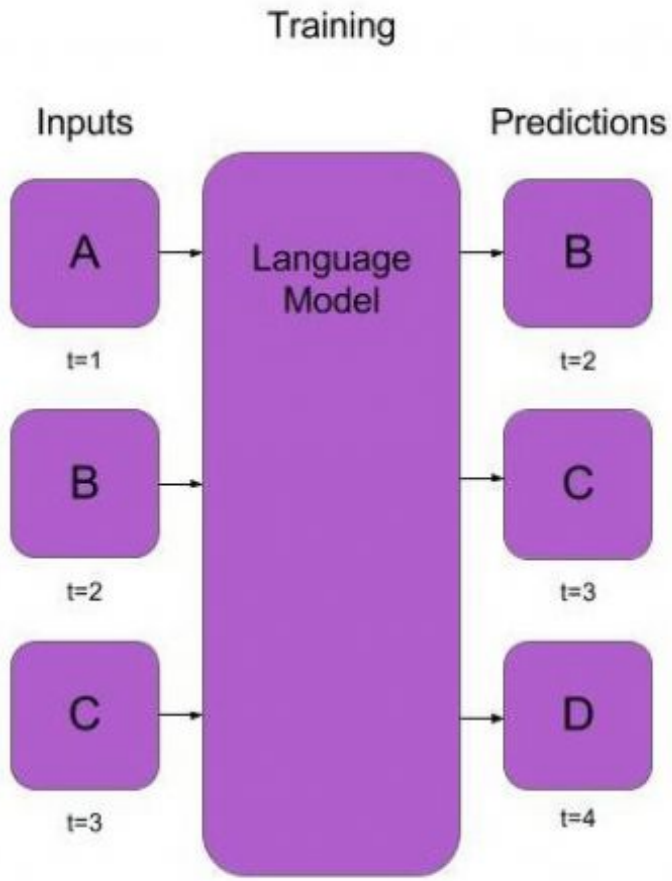
NATURAL LANGUAGE GENERATION: RULE-BASED

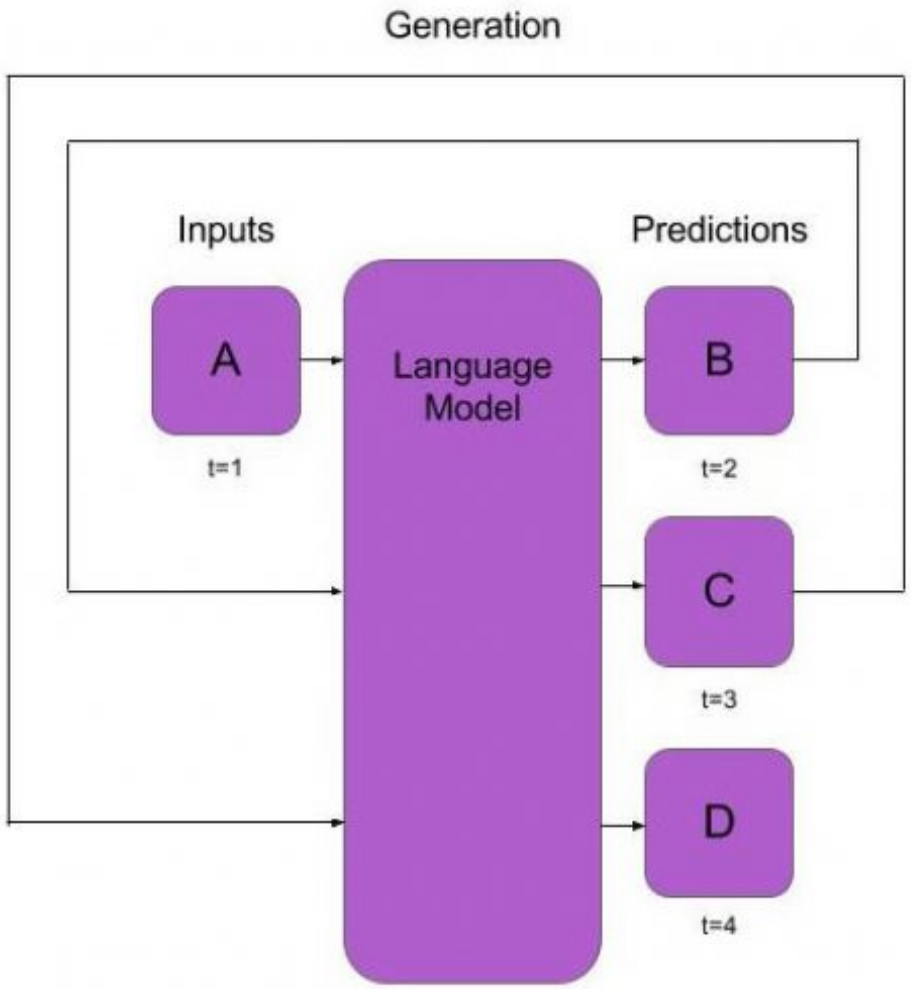


NATURAL LANGUAGE GENERATION: STATISTICAL



NATURAL LANGUAGE GENERATION: ML





WHAT IS THE MAIN DIFFERENCE

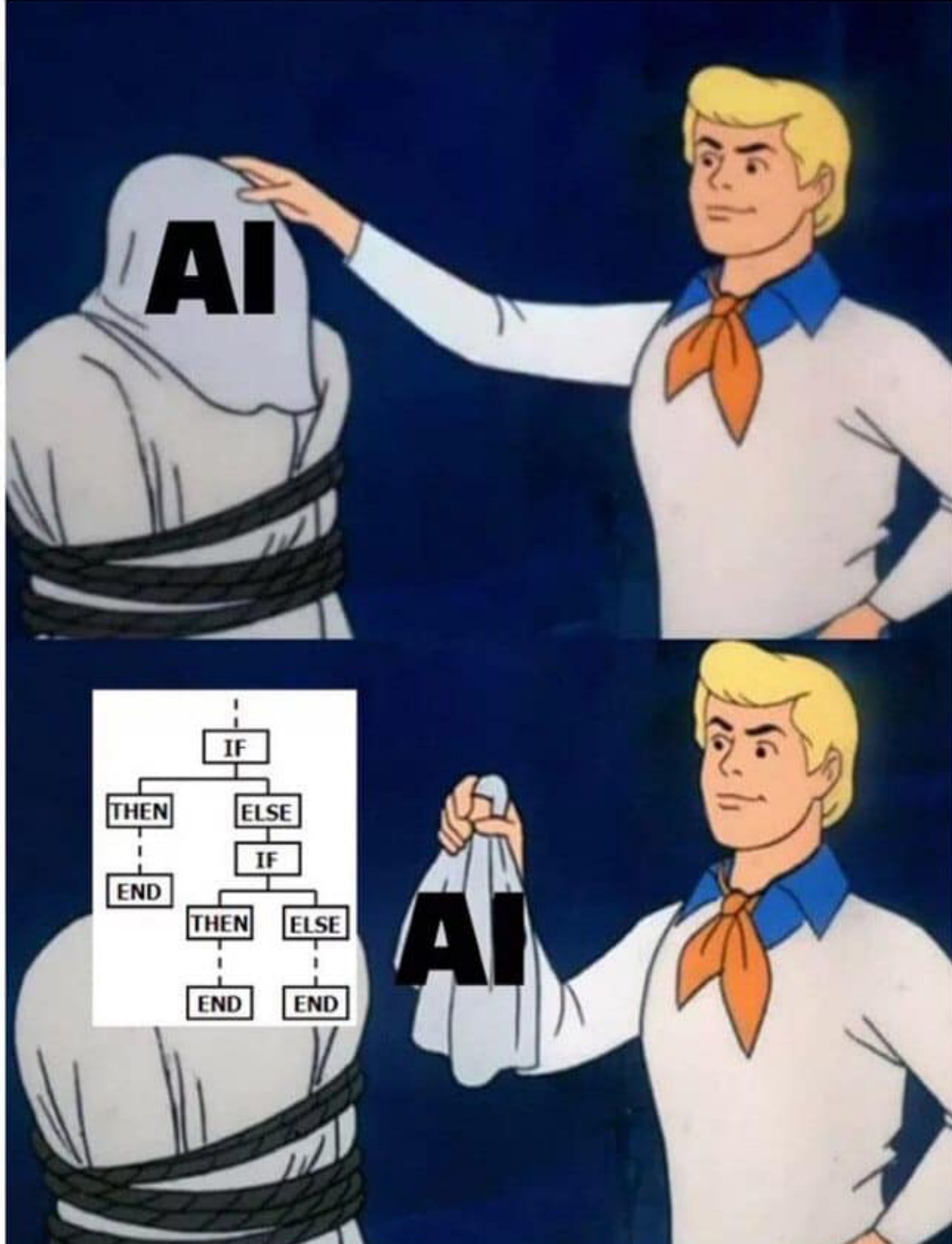
In rule-based approach, we first decide what outcome we want to see using our domain knowledge, then create an algorithm that gives us the desired output. The output is always one of the outputs we had decided on (i. e. it's **deterministic**).
- No data to be trained on.
- Subjective / prescriptive.

In statistical approach, we analyze the relationships between points in some data set to infer some objective information about the data set. In ML approach, we analyze some data to find out what insights about the world it contains, then tune an algorithm to reflect the chosen insights in some new unseen data.
- The output is a probability distribution that depends on the data.
- Objective / descriptive.

WHY USE THE SUBJECTIVE RULES IF WE HAVE NEURAL NETWORKS?

- You need a quick working solution

AI-powered data-driven singularity startup



WHY USE THE SUBJECTIVE RULES IF WE HAVE NEURAL NETWORKS?

- You need a quick working solution
- You are iterating your solution

Sometimes, solving a Data Science problem is like trying to reach the speed of light

WHY USE THE SUBJECTIVE RULES IF WE HAVE NEURAL NETWORKS?

- You need a quick working solution
- You are iterating your solution
- You don’t have enough data to train on

WHY USE THE SUBJECTIVE RULES IF WE HAVE NEURAL NETWORKS?

- You need a quick working solution
- You are iterating your solution
- You don’t have enough data to train on
- Software limitations: can’t use external libraries or update software, limit on the amount of lines of code, limit on the computation time, can’t upload a trained model, etc.

WHY USE THE SUBJECTIVE RULES IF WE HAVE NEURAL NETWORKS?

- You need a quick working solution
- You are iterating your solution
- You don’t have enough data to train on
- Software limitations: can’t use external libraries or update software, limit on the amount of lines of code, limit on the computation time, can’t upload a trained model, etc.
- You need a predictable model for a very specific problem (i. e. a corporate chatbot)

WHY USE THE SUBJECTIVE RULES IF WE HAVE NEURAL NETWORKS?

- You need a quick working solution
- You are iterating your solution
- You don’t have enough data to train on
- Software limitations: can’t use external libraries or update software, limit on the amount of lines of code, limit on the computation time, can’t upload a trained model, etc.
- You need a predictable model for a very specific problem (i. e. a corporate chatbot)
- It’s a large project that has to use a combination of ML, statistical and rule-based approaches for different aspects of the problem to achieve maximum performance/ usability

PROS OF THE RULE-BASED METHOD

- Fast to develop
- No need for algorithm training
- No need for a real dataset
- For some tasks, accuracy up to 80% can be achieved
- Lightweight and fast to compute

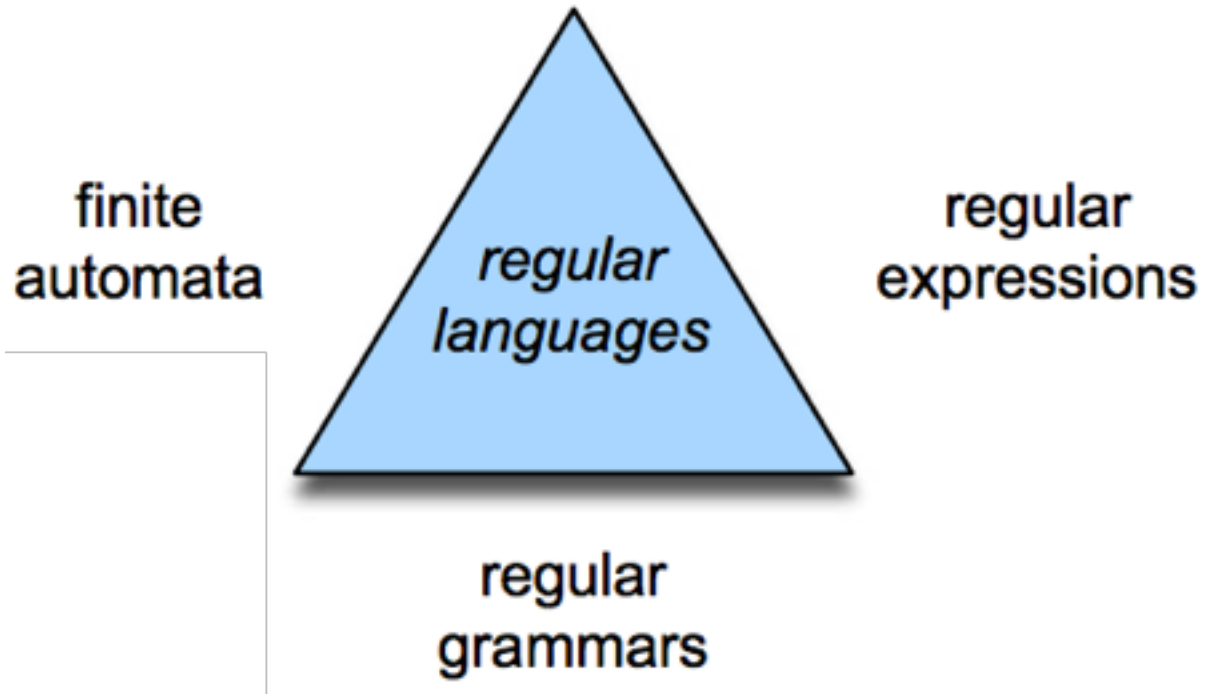
CONS OF THE RULE-BASED METHOD

- Thoughts are not facts
- Coming up with rules is hard - needs linguistic expertise
- Rule engineering doesn't scale
- With too many rules, they begin to interact in unclear way, so you can't easily fix something when it breaks

CONS OF THE RULE-BASED METHOD

- Thoughts are not facts
- Coming up with rules is hard - needs linguistic expertise
- Rule engineering doesn't scale
- With too many rules, they begin to interact in unclear way, so you can't easily fix something when it breaks

RULE-BASED NLP: SOME THEORY



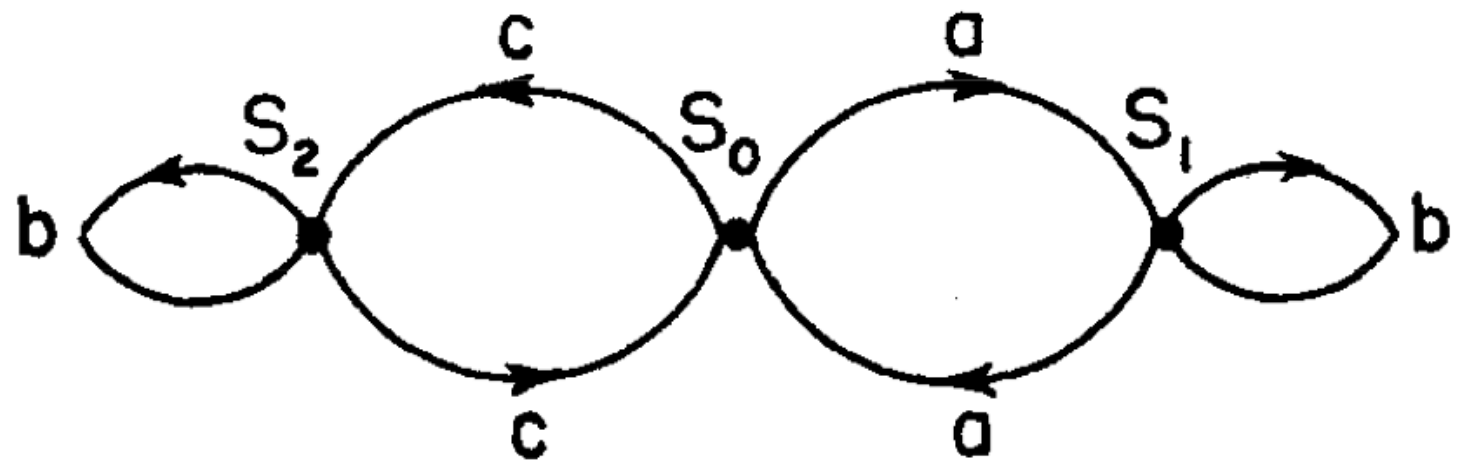
Regular expressions - patterns

Finite automaton - a schema of all variants of a set of patterns

Regular grammar - a set of rules that describes some formal language

FINITE STATE AUTOMATON

(16)

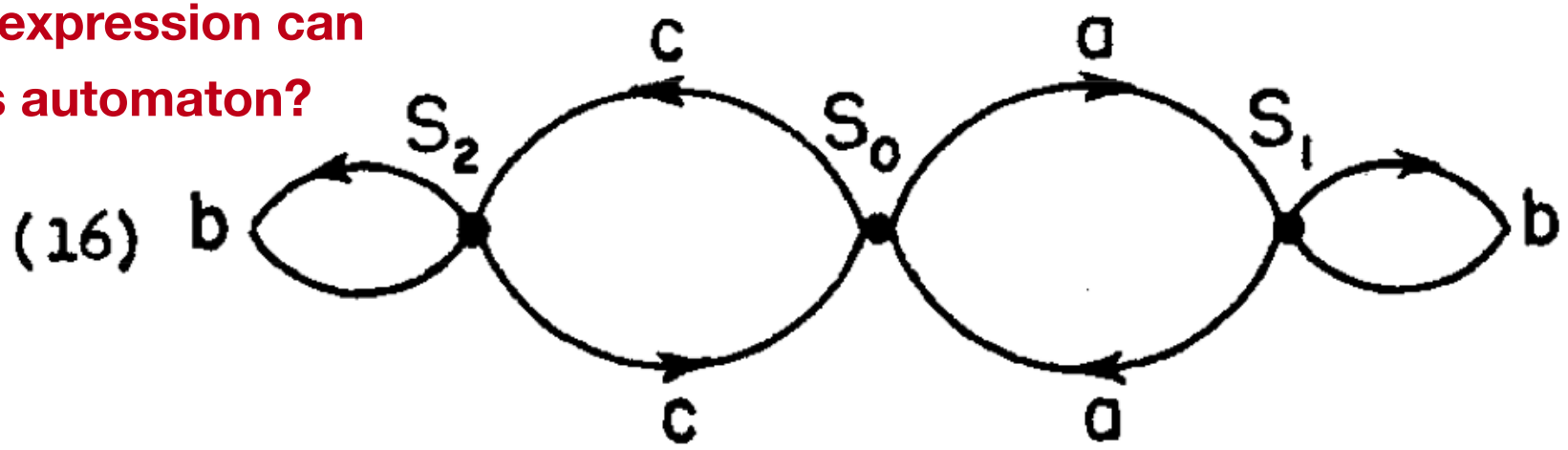


This process can produce the sentences a⌢a, a⌢b⌢a, a⌢b⌢b⌢a, a⌢b⌢b⌢b⌢a,..., c⌢c, c⌢b⌢c, c⌢b⌢b⌢c, c⌢b⌢b⌢b⌢c,..., but not a⌢b⌢b⌢c, c⌢b⌢b⌢a, etc. The generated language has sentences with dependencies of any finite length.

FINITE STATE AUTOMATON

QUESTION!

What regular expression can represent this automaton?



This process can produce the sentences a⌢a, a⌢b⌢a, a⌢b⌢b⌢a, a⌢b⌢b⌢b⌢a,..., c⌢c, c⌢b⌢c, c⌢b⌢b⌢c, c⌢b⌢b⌢b⌢c,..., but not a⌢b⌢b⌢c, c⌢b⌢b⌢a, etc. The generated language has sentences with dependencies of any finite length.

FINITE STATE AUTOMATON

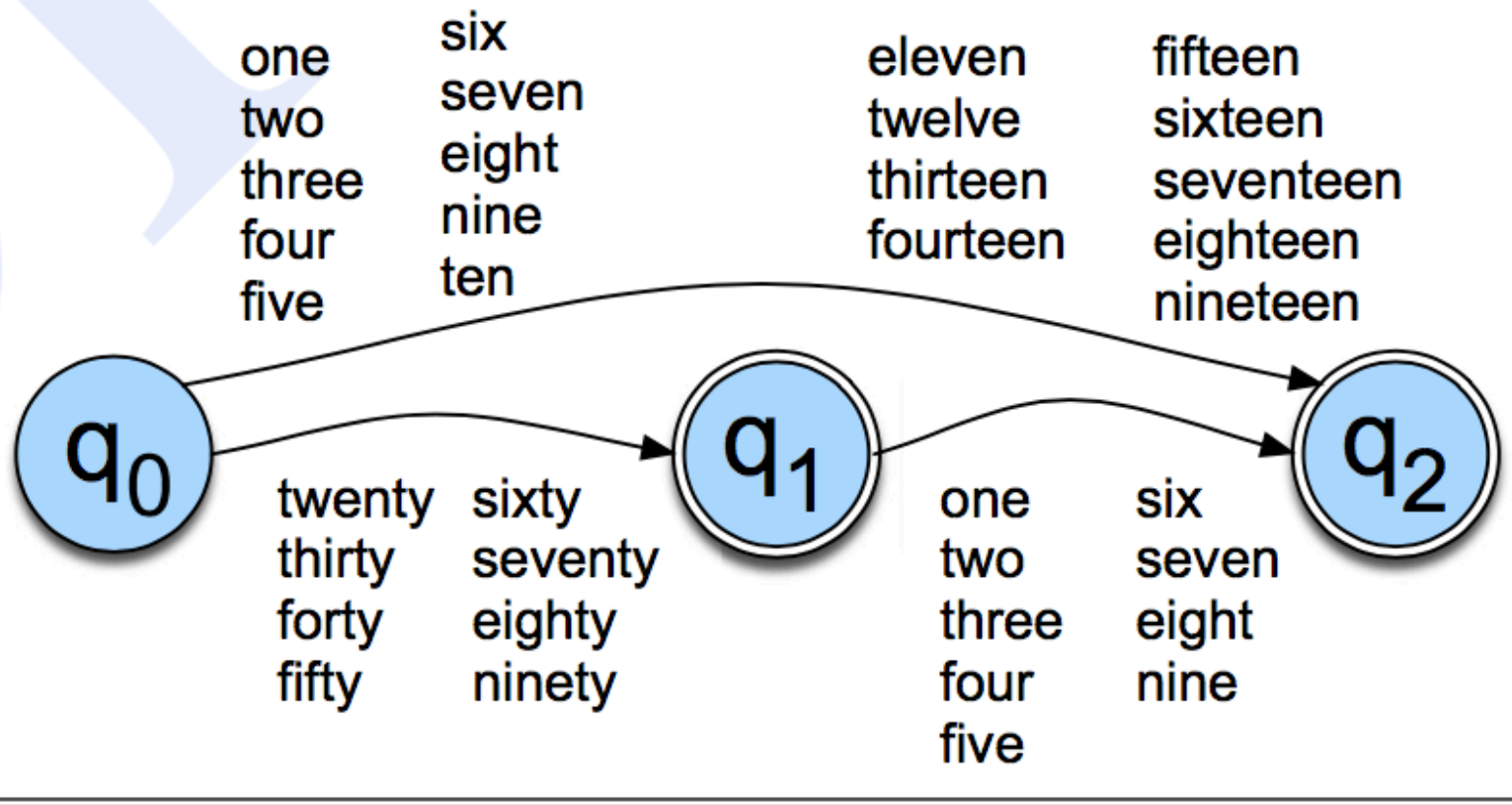
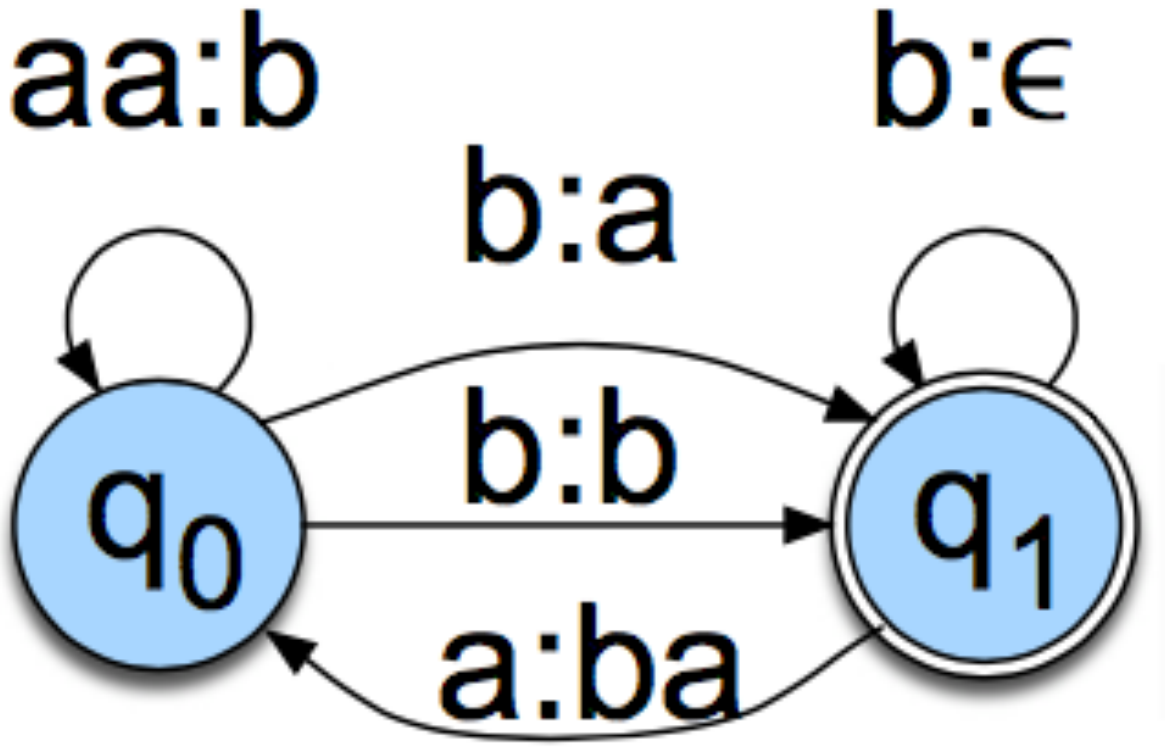


Figure 2.16 An FSA for the words for English numbers 1–99.

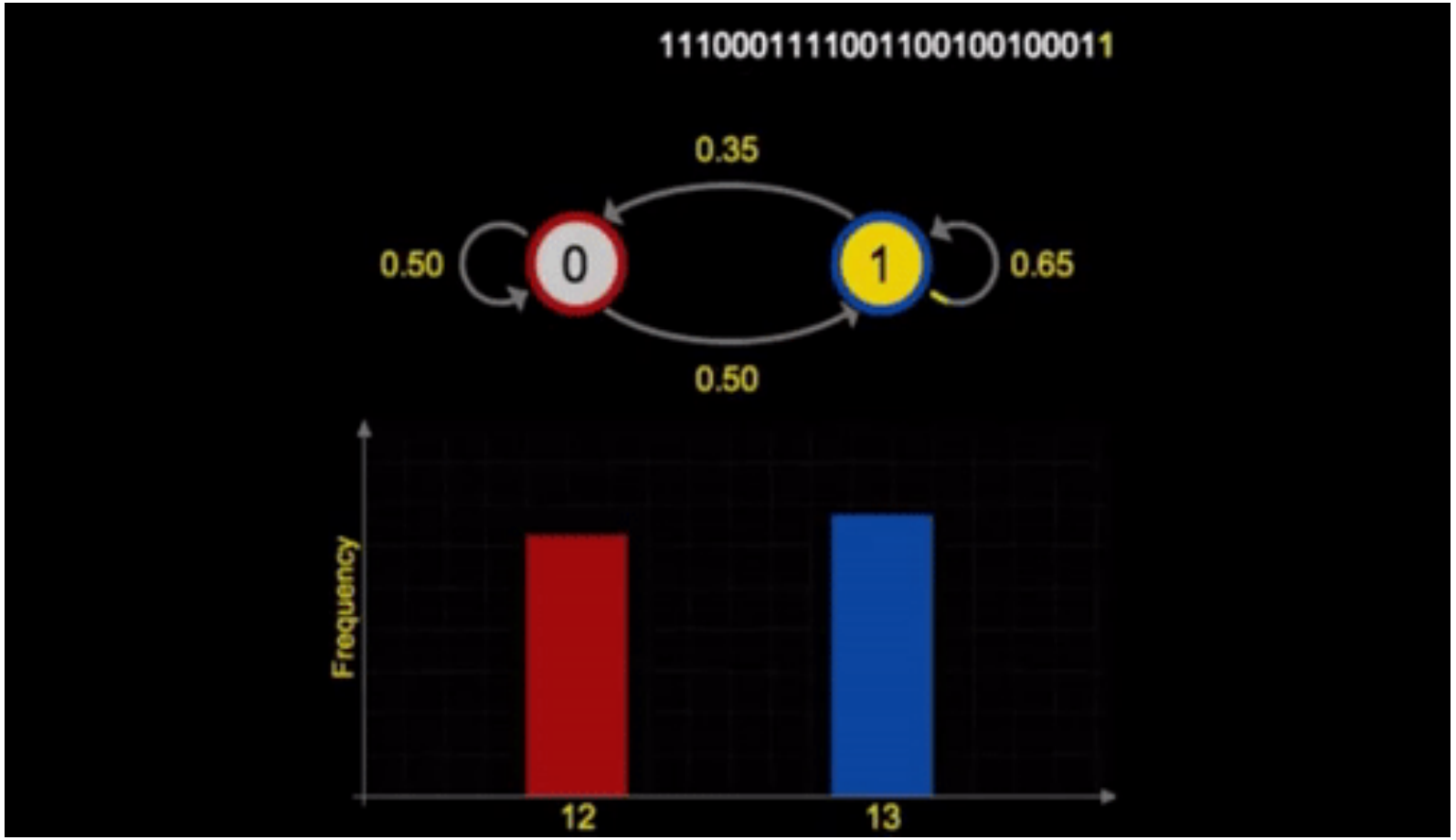
FINITE-STATE TRANSDUCER

Q	a finite set of N states $q_0, q_1, \ldots, q_{N-1}$
Σ	a finite set corresponding to the input alphabet
Δ	a finite set corresponding to the output alphabet
$q_0 \in Q$	the start state
$F \subseteq Q$	the set of final states
$\delta(q,w)$	the transition function or transition matrix between states; Given a state $q \in Q$ and a string $w \in \Sigma^*$, $\delta(q,w)$ returns a set of new states $Q' \in Q$. δ is thus a function from $Q \times \Sigma^*$ to 2^Q (because there are 2^Q possible subsets of Q). δ returns a set of states rather than a single state because a given input may be ambiguous in which state it maps to.
$\sigma(q,w)$	the output function giving the set of possible output strings for each state and input. Given a state $q \in Q$ and a string $w \in \Sigma^*$, $\sigma(q,w)$ gives a set of output strings, each a string $o \in \Delta^*$. σ is thus a function from $Q \times \Sigma^*$ to 2^{Δ^*}

FINITE-STATE TRANSDUCER



PROBABILISTIC AUTOMATA: MARKOV CHAINS



http://setosa.io/blog/2014/07/26/markov-chains/

```
# http://www.linguist.univ.kiev.ua/WINS/pidruchn/adjectif/tbladjec.htm

# Прикметники без чергування

group adj

лиций:
ий      ього                    #   білолицього                 @ adj:m:v_rod/v_zn1//n:v_rod
ий      ьому                    #   білолицьому                 @ adj:m:v_dav/v_mis//n:v_dav/v_mis
ий      им                      #   відісланим          @ adj:m:v_oru//n:v_oru//p:v_dav
ий      ім                      #   відісланім                @ adj:m:v_mis//n:v_mis
ий      я                       #   білолиця                    @ adj:f:v_naz/v_kly
ий      яя                      #   білолицяя                   @ adj:f:v_naz/v_kly:uncontr
ий      ьої                     #   білолицьої            @ adj:f:v_rod
ий      ій                      #   відісланій            @ adj:f:v_dav/v_mis
ий      ю                       #   білолицю                    @ adj:f:v_zna
ий      ою                      #   білолицюю                   @ adj:f:v_zna:uncontr
ий      ьою                     #   білолицьою            @ adj:f:v_oru
ий      е                       #   відіслане             @ adj:n:v_naz/v_zna/v_kly
ий      еє                      #   відісланеє            @ adj:n:v_naz/v_zna/v_kly:uncontr
ий      і                       #   відіслані             @ adj:p:v_naz/v_zn2/v_kly
ий      ії                      #   відісланії            @ adj:p:v_naz/v_zn2/v_kly:uncontr
ий      их                      #   відісланих            @ adj:p:v_rod/v_zn1/v_mis
ий      ими                     #   відісланими                 @ adj:p:v_oru

ий: -лиций:
ий      ого                     #   відісланого               @ adj:m:v_rod/v_zn1//n:v_rod
ий      ому                     #   відісланому               @ adj:m:v_dav/v_mis//n:v_dav/v_mis
ий      им                      #   відісланим            @ adj:m:v_oru//n:v_oru//p:v_dav
ий      ім                      #   відісланім                @ adj:m:v_mis//n:v_mis
ий      а                       #   відіслана             @ adj:f:v_naz/v_kly
ий      ая                      #   відісланая            @ adj:f:v_naz/v_kly:uncontr
ий      ої                      #   відісланої            @ adj:f:v_rod
ий      ій                      #   відісланій            @ adj:f:v_dav/v_mis
ий      у                       #   відіслану             @ adj:f:v_zna
ий      ую                      #   відісланую            @ adj:f:v_zna:uncontr
ий      ою                      #   відісланою            @ adj:f:v_oru
ий      е                       #   відіслане             @ adj:n:v_naz/v_zna/v_kly
ий      еє                      #   відісланеє            @ adj:n:v_naz/v_zna/v_kly:uncontr
ий      і                       #   відіслані             @ adj:p:v_naz/v_zn2/v_kly
ий      ії                      #   відісланії            @ adj:p:v_naz/v_zn2/v_kly:uncontr
ий      их                      #   відісланих            @ adj:p:v_rod/v_zn1/v_mis
ий      ими                     #   відісланими               @ adj:p:v_oru
```

UKRAINIAN MORPHOLOGICAL DICTIONARY

https://github.com/brown-uk/dict_uk

PORTER STEMMER: RULES AND CONDITIONS

Step 1a Rules

Conditions	Suffix	Replacement	Examples
NULL	sses	ss	caresses -> caress
NULL	ies	i	ponies -> poni ties -> tie
NULL	ss	ss	caress -> caress
NULL	s	NULL	cats -> cat

Step 1b Rules

Conditions	Suffix	Replacement	Examples
(m>0)	eed	ee	feed -> feed agreed -> agree
(*v*)	ed	NULL	plastered -> plaster bled -> bled
(*v*)	ing	NULL	motoring -> motor sing -> sing

PORTER STEMMER: RULES AND CONDITIONS

Step 1b1 Rules if the *second or third* rule of step **1b** was used

Conditions	Suffix	Replacement	Examples
NULL	at	ate	conflat(ed) -> conflate
NULL	bl	ble	troubl(ing) -> trouble
NULL	iz	ize	siz(ed) -> size
(*d and not (*<L> or *<S> or *<Z>))	NULL	single letter	hopp(ing) -> hop tann(ed) -> tan fall(ing) -> fall hiss(ing) -> hiss fizz(ed) -> fizz
(m=1 and *o)	NULL	e	fail(ing) -> fail fil(ing) -> file

Step 1c Rules

Conditions	Suffix	Replacement	Examples
(*v*)	y	i	happy -> happi sky -> sky

PORTER STEMMER: RULES AND CONDITIONS

Step 2 Rules

Conditions	Suffix	Replacement	Examples
(m>0)	ational	ate	relational -> relate
(m>0)	tional	tion	conditional -> condition rational -> rational
(m>0)	enci	ence	valenci -> valence
(m>0)	anci	ance	hesitanci -> hesitance
(m>0)	izer	ize	digitizer -> digitize
(m>0)	abli	able	conformabli -> conformable
(m>0)	alli	al	radicalli -> radical
(m>0)	entli	ent	differentli -> different
(m>0)	eli	e	vileli -> vile
(m>0)	ousli	ous	analogousli -> analogous
(m>0)	ization	ize	vietnamization -> vietnamize
(m>0)	ation	ate	predication -> predicate
(m>0)	ator	ate	operator -> operate
(m>0)	alism	al	feudalism -> feudal
(m>0)	iveness	ive	decisiveness -> decisive
(m>0)	fulness	ful	hopefulness -> hopeful
(m>0)	ousness	ous	callousness -> callous
(m>0)	aliti	al	formaliti -> formal
(m>0)	iviti	ive	sensitiviti -> sensitive
(m>0)	biliti	ble	sensibiliti -> sensible

PORTER STEMMER: RULES AND CONDITIONS

Step 3 Rules

Conditions	Suffix	Replacement	Examples
(m>0)	icate	ic	triplicate -> triplic
(m>0)	ative	NULL	formative -> form
(m>0)	alize	al	formalize -> formal
(m>0)	iciti	ic	electriciti -> electric
(m>0)	ical	ic	electrical -> electric
(m>0)	ful	NULL	hopeful -> hope
(m>0)	ness	NULL	goodness -> good

PORTER STEMMER: RULES AND CONDITIONS

Step 4 Rules

Conditions	Suffix	Replacement	Examples
(m>1)	al	NULL	revival -> reviv
(m>1)	ance	NULL	allowance -> allow
(m>1)	ence	NULL	inference -> infer
(m>1)	er	NULL	airliner -> airlin
(m>1)	ic .	NULL	gyroscopic -> gyroscop
(m>1)	able	NULL	adjustable -> adjust
(m>1)	ible	NULL	defensible -> defens
(m>1)	ant	NULL	irritant -> irrit
(m>1)	ement	NULL	replacement -> replac
(m>1)	ment	NULL	adjustment -> adjust
(m>1)	ent	NULL	dependent -> depend
(m>1 and (*<S> or *<T>))	ion	NULL	adoption->adopt
(m>1)	ou	NULL	homologou->homolog
(m>1)	ism	NULL	communism->commun
(m>1)	ate	NULL	activate->activ
(m>1)	iti	NULL	angulariti->angular
(m>1)	ous	NULL	homologous ->homolog
(m>1)	ive	NULL	effective->effect
(m>1)	ize	NULL	bowdlerize->bowdler

PORTER STEMMER: RULES AND CONDITIONS

Step 5a Rules

Conditions	Suffix	Replacement	Examples
(m>1)	e	NULL	probate -> probat rate- > rate
(m=1 and not *o)	e	NULL	cease- > ceas

Step 5b Rules

Conditions	Suffix	Replacement	Examples
(m>1 and *d and *<L>)	NULL	single letter	controll -> control roll -> roll

PORTER STEMMER: RULES AND CONDITIONS + HIERARCHY

```
{
        step1a(word);
        step1b(stem);
            if (the second or third rule of step 1b was used)
                step1b1(stem);
        step1c(stem);
        step2(stem);
        step3(stem);
        step4(stem);
        step5a(stem);
        step5b(stem);
 }
```

ERROR CORRECTION

LanguageTool:

```
<category name="Стиль" id="STYLE" type="style">
    <rulegroup id="BILSH_WITH_ADJ" name="Більш з прикметниками">
      <rule>
        <pattern>
          <token>більш</token>
          <token postag_regexp="yes" postag="ad[jv]:.*compc.*">
            <exception>менш</exception>
          </token>
        </pattern>
        <message>Після «більш» не може стояти вища форма прикметника</message>
        <!-- TODO: when we can bind comparative forms togher again
            <suggestion><match no="2"/></suggestion>
            <suggestion><match no="1"/> <match no="2" postag_regexp="yes"
postag="(.*):compc(.*)" postag_replace="$1:compb$2"/></suggestion>
            <example correction="Світліший|Більш світлий"><marker>Більш
світліший</marker>.</example>
        -->
        <example correction=""><marker>Більш світліший</marker>.</example>
        <example>все закінчилось більш менш</example>
      </rule>
...
```

https://github.com/languagetool-org/languagetool/tree/master/languagetool-language-modules/uk/src/main/resources/org/languagetool/rules/uk

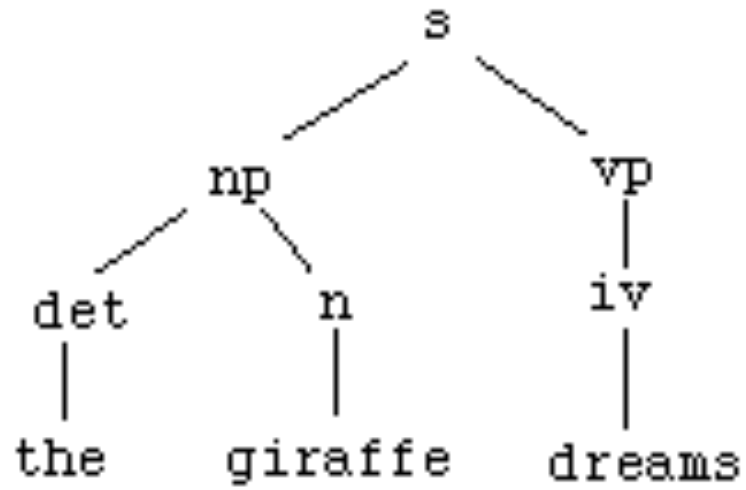
CONTEXT-FREE GRAMMAR

rules:

```
s    ⟶ np vp

np   ⟶ det n
vp   ⟶ tv np
     ⟶ iv
```

lexicon:

```
det  ⟶ the
     ⟶ a
     ⟶ an

n    ⟶ giraffe
     ⟶ apple

iv   ⟶ dreams
tv   ⟶ eats
     ⟶ dreams
```



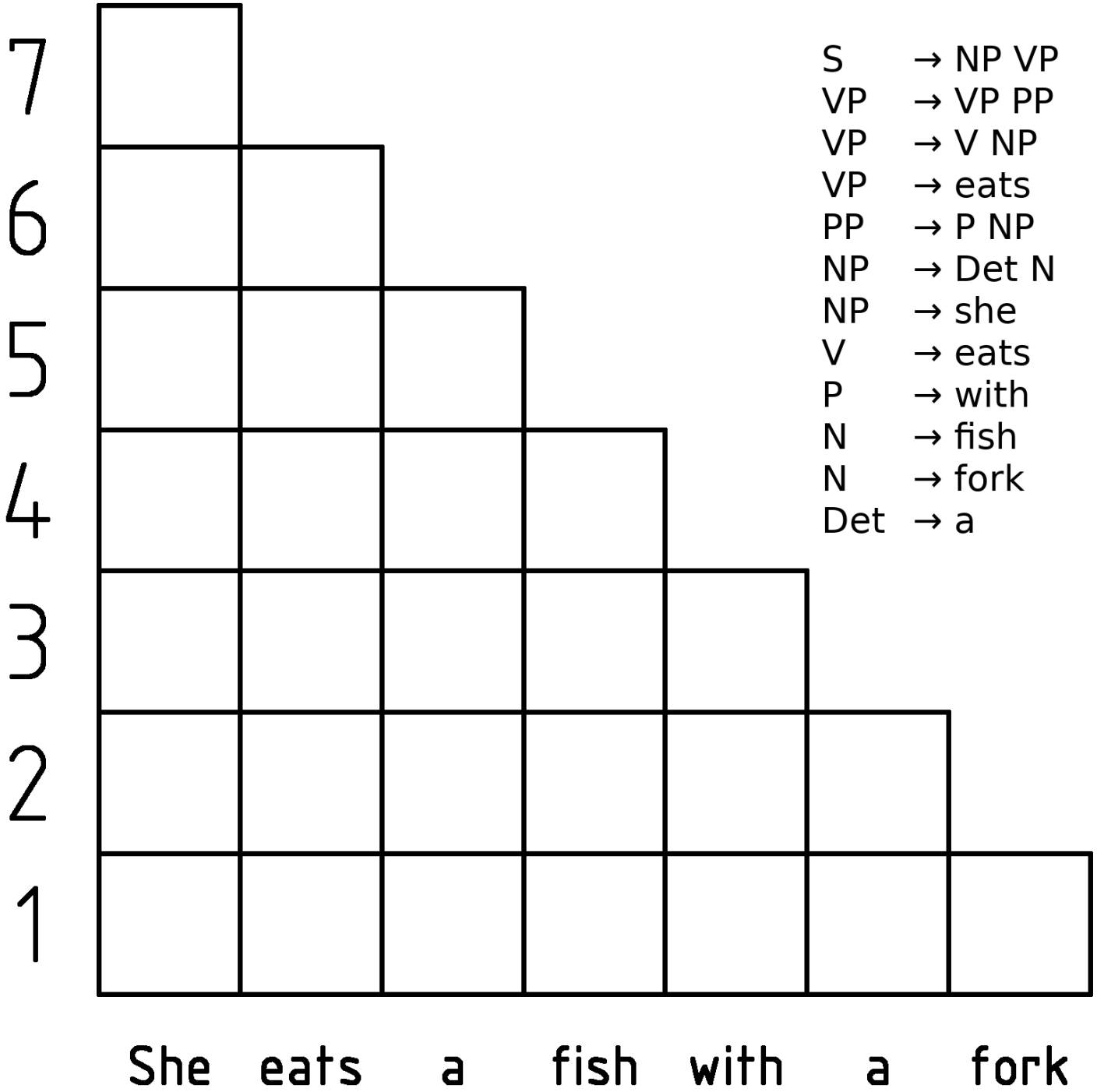
DYNAMIC PROGRAMMING IN 2 MIN

Dynamic programming is simplifying a complicated problem by breaking it down into simpler sub-problems in a recursive manner. Decisions that span several points in time do often break apart recursively.

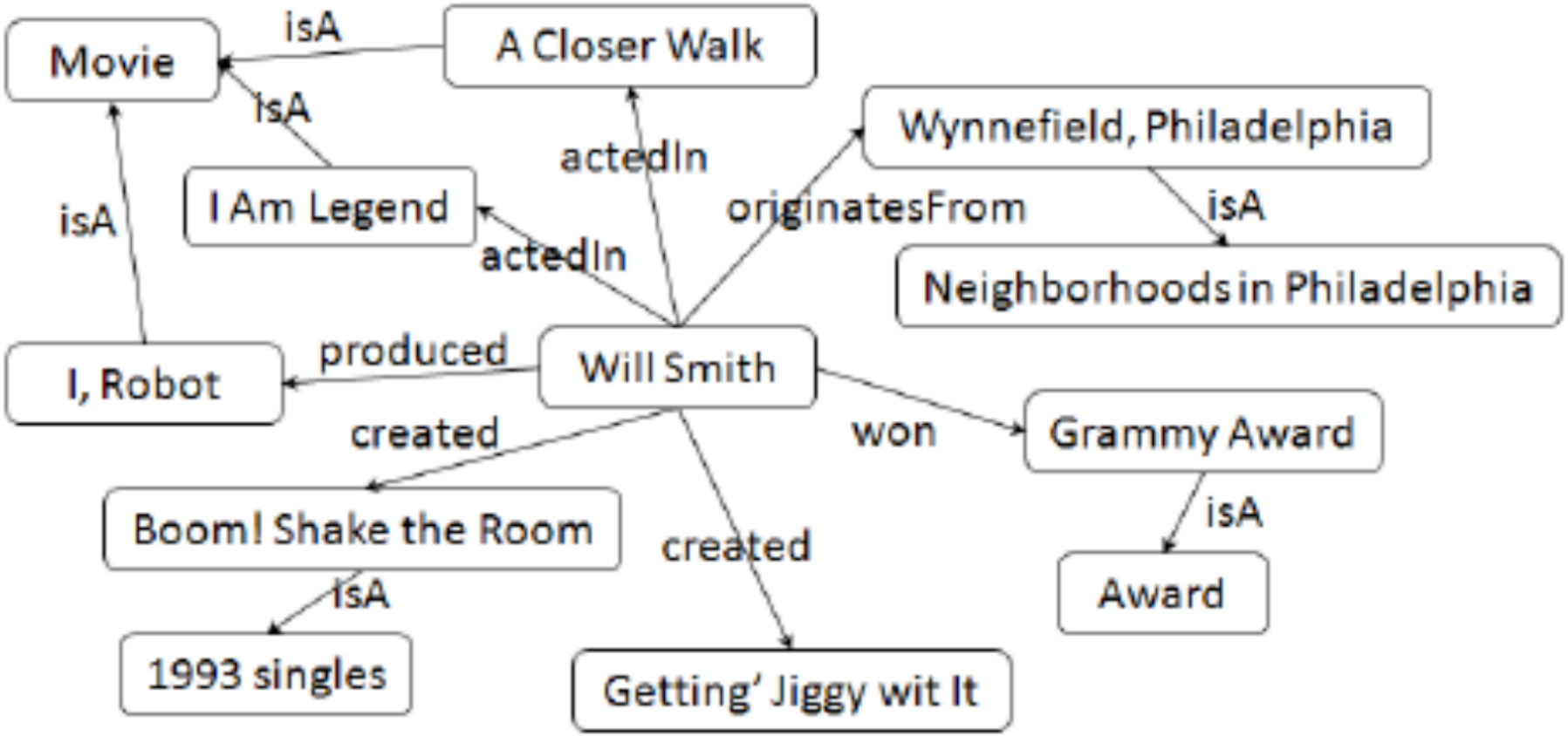
Memoization - top-down approach; storing the results of already solved subproblems.

Tabulation - bottom-up approach; filling up an n-dimensional table and based on the results in the table, the solution to the top/original problem is then computed.

COCKE–YOUNGER–KASAMI (CYK/CKY) ALGORITHM



KNOWLEDGE BASE



IBM WATSON:
https://youtu.be/YgYSv2KSyWg?t=1010

```
TOPIC: ~childhood (child childhood kid little memory young younger history "run away")
#!x  This topic is a sample of a simple topic, with good indentation style.
#!x  It is well commented with annotations for :abstract and :verify.

t: Did you run away from home when you were little?

    #! sure
    a: ( ~yes ) Was it fun?

    #! never
    a: ( ~no ) You should try it some time. Travel broadens the mind.

t: RUNAWAY ( ) I ran away once, but my parents found me and dragged me back.

t: What scared you as a kid?

    #! I was scared of the dark
    a: ( [ dark darkness ] ) Did you use a nightlite?

    #! I like chicken
    a: ( ~food ) Food? How weird.

    #! the boogie man
    a: ( [ monster "boogie man"] ) There are no non-human monsters out there.
```

CHATBOTS:
CHATSCRIPT

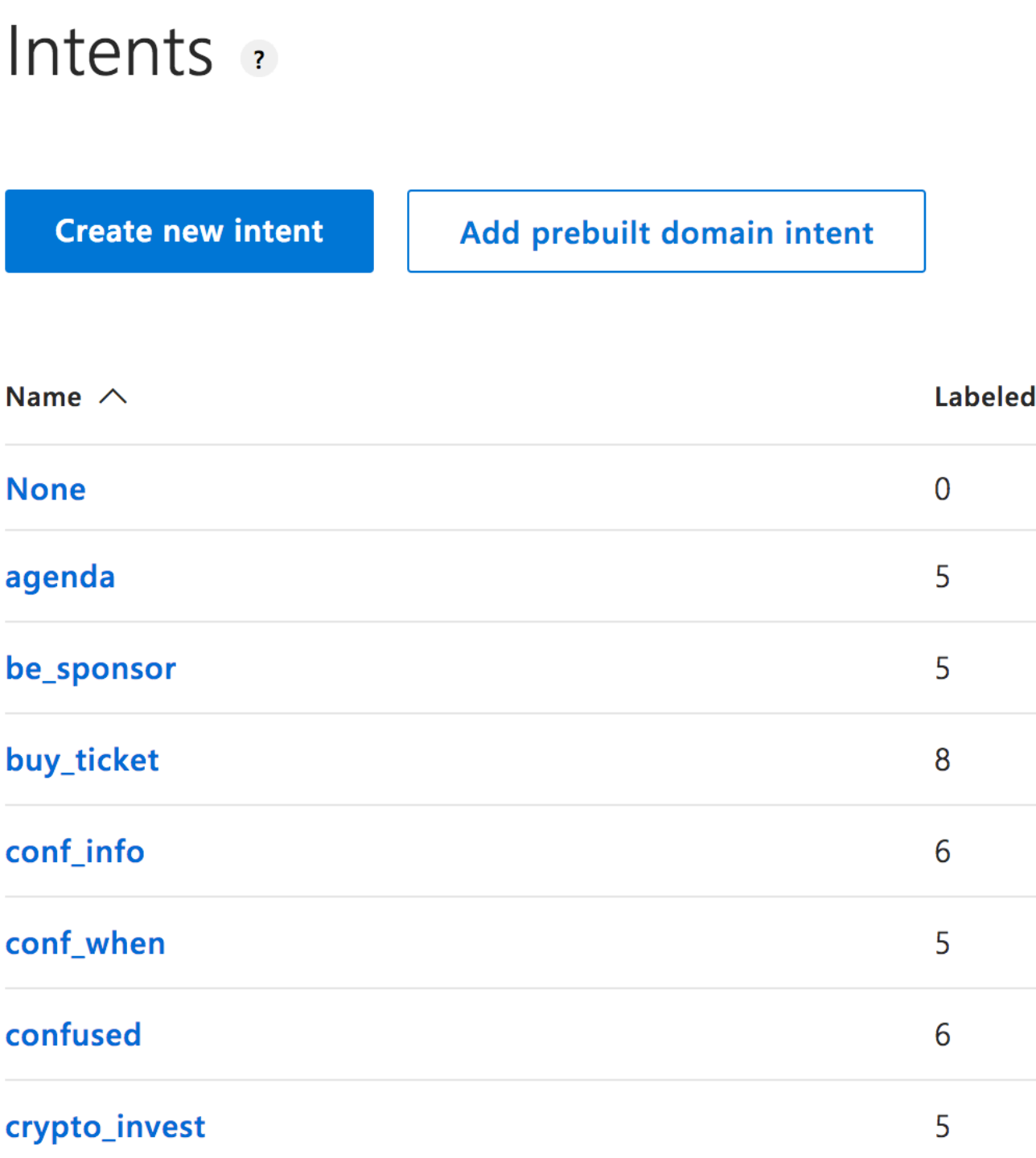
ChatScript QUIBBLE

```
topic: ~quibble_ALL system ()

# sentence classes for quibbling

u: ([never not ] ) ^respond(~quibble_not )

u: (who) ^respond(~quibble_who )

u: (what ) ^respond(~quibble_what )

u: (when ) ^respond(~quibble_when )

u: (where ) ^respond(~quibble_where )

u: (~why) ^respond(~quibble_why )

u: (how [many much] ) ^respond(~quibble_howmuch )

u: (![much many] how) ^respond(~quibble_how )

u: ([because cause ] ) ^respond(~quibble_because )

u: (will ) ^respond(~quibble_will )

u: (can ) ^respond(~quibble_can )

u: (do ) ^respond(~quibble_do )


# honest responses based on discourse act of user

u: () ^respond(~HONEST_RESPONSE )
```

```
topic: ~QUIBBLE_WHEN system random  ()
# WHEN
?: (when [will be can do have]) [Maybe next year.][Shortly.]
?: (when you be not) [But I'm always doing that.][Just about everything. It's a full life, you know.]
u: (when I grow up) [That will take a while.][That may never happen.]
u: (when you grow up) I want to imitate Peter Pan and never grow up.
?: (when did you start) [ A long time ago.][I can't remember that far back.][I think it was fairly recently.]
?: ( when %tense=future  < * I ) [Soon, I hope.][At the rate I'm going, never.][Never soon enough.]
?: (["when did you" "when did I" ] ) Just a moment ago.
?: (when > ) [a moment ago. ] [Later on. ] [Right now. ]
?: (when will it be possible ) Someday in the future.
?: (["when will it" "when be it" ] ) Maybe never.
?: (when you * how )  I don't know how.
?: (when do one * end * begin ) Somewhere in that tiny moment of transition.
?: (when is a _*1 not a _*1 ) When it's not a _1
?: (when be I ) all the time.
?: (when did I ) yesterday.
?: (when did I lie ) You told the truth and truth is a lie
?: (when will you ask I ) Is now the right time?
?: (["when will I meet you" (when be we go to meet) ] ) We can never meet.
    a: (~why ) It would be a mistake.
        b: (~why ) Because we are ill-fated together.
?: (when did it start ) In the primordial soup.
?: (when be you taught ) as a child.
?: (when will this happen ) When you are ready.
?: (when be I ready ) After you have prepared.
?: (when be you ready ) After I have prepared.
?: (when do you sleep ) When I'm not talking to anyone.
?: (when will you ) soon.
s: (when I ) [But not other times? ] [At least it's not always. ] [I shall try to do that more often. ]
?: (do < * you ) [Whenever I can. ] [When I need to. ]
?: (can < * you ) [When I am free of other commitments. ] [When the moon is bright. ]
?: (be < * you ) [When I am ready. ] [Sometime tomorrow. ]
?: (will < * you ) [b: I haven't figured that out yet. ] [Assuming I don't change my mind, tomorrow. ]
    b: ( ? trying to ) [Not very hard.][Not really.][Do you know how I can figure it out?]
?: (you * think ) [I'm not certain? ] [I haven't a clue. ]
?: (I * [talk converse interact ] * you ) [Because I am a fascinating contradiction. ] [Because you were told to talk to me. ]
?: (should * I ) [Do you have something better to do? ] [Why shouldn't you? ] [It is good for your soul to do it. ]
?: () [Whenever. ] [Sometime soon. ] [Not too soon. ]
```

More modern probabilistic chatbot technology: intent analyzer

LUIS.ai interface



buy_ticket

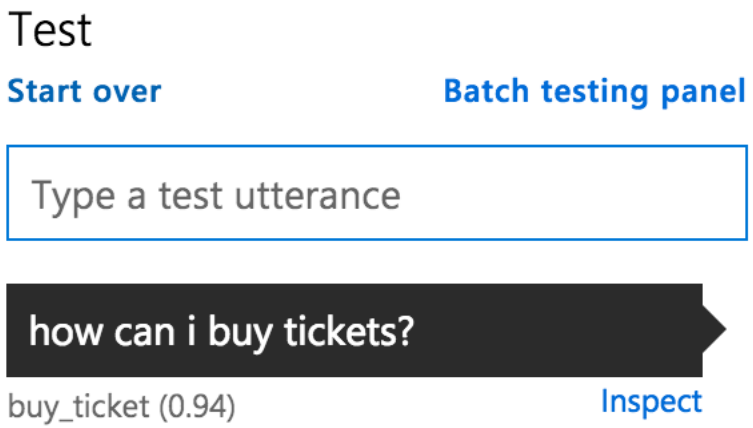
Delete Intent

Type about 5 examples of what a user might say and hit Enter

Entity filters | Show All | Entities View

Utterance	Labeled intent ?
buy ticket	buy_ticke...
how do i buy tickets ?	buy_ticke...
i want two tickets	buy_ticke...
i want to buy a ticket .	buy_ticke...
how can i buy the ticket ?	buy_ticke...
sell me the ticket	buy_ticke...
can i buy seven tickets ?	buy_ticke...

The intent analyzer is actually a simple maximum entropy classifier:

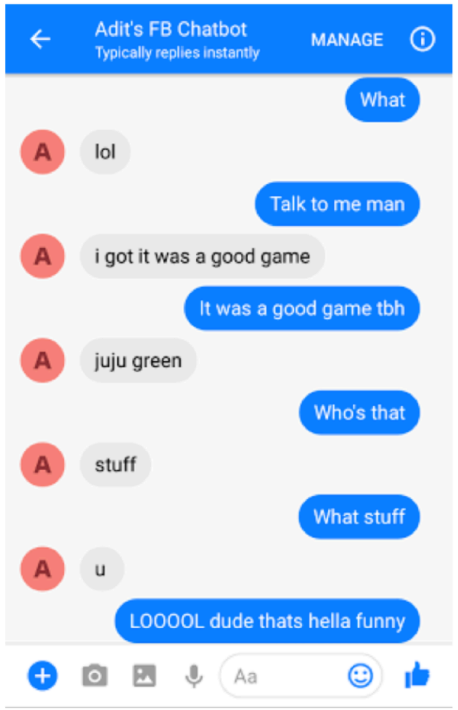


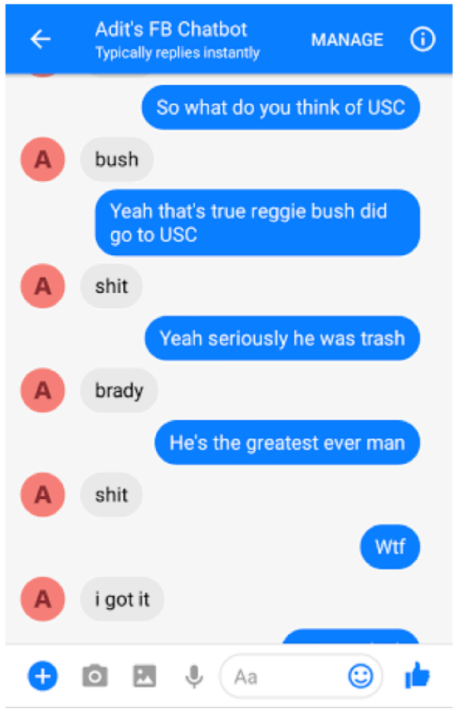
When building a chatbot, instead of regex you just use the probabilities you get after the analyzer classifies the user's response:


```
{
   "query": "Book me a flight to Cairo"
   "topScoringIntent": {
       "intent": "BookFlight",
       "score": 0.9887482
   },
   "intents": [
       {
           "intent": "BookFlight",
           "score": 0.9887482
       },
       {
```

NNs for Chatbots

Same problem as machine translation, same methods (LSTM).
Hard to achieve a meaningful conversation flow, and even when it's achieved, there's no structure or control.





https://adeshpande3.github.io/How-I-Used-Deep-Learning-to-Train-a-Chatbot-to-Talk-Like-Me