Yet another *Artificial Intelligence 2* Summary

Written by Philip K.[*]

Last updated for the Summer Semester 2021

6 Probability Theory

Def. 86 (Probability Model $\langle \Omega, P \rangle$)**.** consists of a countable sample space Ω and a probability function $P : \Omega \to [0;1]$, s.t. $\sum_{\omega \in \Omega} P(\omega) = 1$

Def. 87 (Event)**.** When a random variable X takes on a value x.

Def. 88 (Conditional/Posterior Probability)**.** The probability

$$P(a\,|\,b) = \frac{P(a \wedge b)}{P(b)},$$

i.e. the chance that event "a" takes place, given the event "b".

Def. 89 (Conditional Independence)**.** Two events a and b are conditionally independent, if $P(a \wedge b\,|\,c) = P(a\,|\,c)P(b\,|\,c)$.

Def. 90 (Probability Distribution)**.** A vector for $\mathbf{P}(X)$ relating each element of the sample space to a probability:

$$\langle P(\omega_1),\dots,P(\omega_n)\rangle.$$

Related concepts:

Joint PD Given $Z \subseteq \{X_1, \dots, X_n\}$, results in a array the probabilities of all events.

Full joint PD Joint PD for all random variables.

Conditional PD Given X and Y, results in a table for every probability $P(X\,|\,Y)$.

Def. 91 (Product Rule)**.** $P(a \wedge b) = P(a\,|\,b)P(b)$

Def. 92 (Chain Rule)**.** Extension of the product rule,

$$P(X_1,\dots,X_n) = P(X_n\,|\,X_{n-1},\dots,X_1)\dots P(X_2|X_1)P(X_1)$$

Def. 93 (Marginalisation)**.** $\mathbf{P}(\mathbf{X}) = \sum_{y \in \mathbf{Y}} \mathbf{P}(\mathbf{X}, y)$

Def. 94 (Normalisation)**.** Given $\mathbf{P}(X\,|\,e)$, and a normalization constant

$$\alpha = \frac{1}{P(x_1\,|\,e) + \dots + P(x_n\,|\,e)},$$

normalization scales each element of the probability distribution s.t. $\sum \alpha \mathbf{P}(X\,|\,e) = 1$

Def. 95 (Bayes' Rule)**.** Given two propositions a and b,

$$P(a\,|\,b) = \frac{P(b\,|\,a)P(a)}{P(b)},$$

where $P(a) \neq 0$ and $P(b) \neq 0$.

Def. 96 (Naive Bayes' Model)**.** In this model, the full joint probability distribution is

$$\mathbf{P}(c\,|\,e_1,\dots,e_n) = \mathbf{P}(c)\prod_i \mathbf{P}(e_i\,|\,c),$$

i.e. a *single* cause c influences a number of cond. independent effects e_i.

[*] `https://gitlab.cs.fau.de/oj14ozun/ai2-summary`, the source for this document should be accessible as a PDF attachment.

6.1 Bayesian Networks

Def. 97 (Bayesian Network)**.** A directed, acyclic graph, where each node corresponds to a random variable, connected by links designating "parent" variables.

A "diagnostic" link points from cause to effect, a "causal" from effect to cause.

Def. 98 (Conditional Probability Table)**.** A table specifying the probability for each node of a Bayesian network given the values of the parent variables.

Def. 99 (Constructing Bayesian Network)**.** Given any fixed order of variables $X_1,\dots,X_n$ a Bayesian network can be constructed by iteratively finding a minimal set $\text{Parent}(X_i) \subseteq \{X_1,\dots,X_i\}$ s.t. $\mathbf{P}(X_i \mid X_{i-1}, \dots, X_1) = \mathbf{P}(X_i \mid \text{Parent}(X_i))$, linking X_i with every $X_j \in \text{Parent}(X_i)$ and associating a conditional probability table with X_i corresponding to $\mathbf{P}(X_i\,|\,\text{Parent}(X_i))$.

Def. 100 (Probabilistic Inference Task)**.** Given a Bayesian network $\mathcal{B}$, calculating $\mathbf{P}(\mathbf{X}\,|\,\mathbf{e})$, where $\mathbf{X}$ is a set of "query variables" and $\mathbf{E}$ is a set of "evidence variables" assigned by an event $\mathbf{e}$. The remaining variables $\mathbf{Y}$ are referred to as "hidden".

This problem can by solved using "Inference by Enumeration":

1. Normalize and marginalize:
$$\mathbf{P}(X|\mathbf{e}) = \alpha \begin{cases} \mathbf{P}(X,\mathbf{e}) & \text{if } \mathbf{Y} = \emptyset \\ \sum_{\mathbf{y} \in \mathbf{Y}} \mathbf{P}(X,\mathbf{e},\mathbf{y}) & \text{otherwise} \end{cases}$$
2. Chain rule, by ordering $X_1,\dots,X_n$ to be consistent with $\mathbf{B}$:
$$\mathbf{P}(X_1,\dots,X_n) = \mathbf{P}(X_n\,|\,X_{n-1},\dots,X_1)\cdots\mathbf{P}(X_2\,|\,X_1)\mathbf{P}(X_1)$$
3. Exploit conditional independence:
$$P(X_i\,|\,X_{i-1},\dots,X_1) = P(X_i\,|\,\text{Parents}(X_i))$$

Alternative methods such as "Variable Elimination" avoid redundant computations by using dynamic programming.

7 Making Simple Decisions Rationally

Def. 101 (Expected Utility)**.** Given a state-evaluation function $U : S \mapsto \mathbb{R}^+$, the expected utility of an action a given evidence $\mathbf{e}$ is

$$EU(a\,|\,\mathbf{e}) = \sum_{s' \in S} P(R(a) = s'\,|\,a,\mathbf{e})U(s'),$$

that rational agents attempt to maximize, where $R(a)$ is the result of the action a.

Def. 102 (Preference)**.** Given two states, one can say that an agent prefers A over B ($A \succ B$), is indifferent ($A \sim B$) or does not prefer B over A ($A \succeq B$).

A preference is rational if orderability, transitivity, continuity, substitutability, monotonicity and decomposability all hold for the relation. If given, there must by a utility function U with

$$U(A) \geq U(B) \iff A \succ B$$

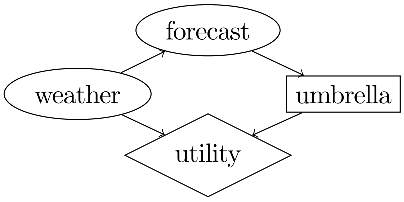
and

$$U([p_1,S_1;\dots;p_n,S_n]) = \sum_i p_i U(S_i),$$

for states S_i and probabilities p_i (Ramsey's Theorem).

Def. 103 (Micromort)**.** A unit of utility, describing a $1/10^{-6}$ chance of death.

Def. 104 (Decision Network)**.** A Bayesian network with "action" and "utility" nodes, to aid with rational decisions. An example (here "umbrella" is an action and "utility" is an utility node):



To decide, maximize the expected utility for the utility nodes by comparing every value for action nodes.

Def. 105 (Value of perfect Information)**.** For a random variable F over D, the VPI given evidence E is

$$\mathrm{VPI}_E(F) := \left(\sum_{f \in D} P(F=f \mid E) EU(\alpha_f \mid E, F=f) \right) - EU(\alpha \mid E),$$

where

$$EU(\alpha \mid E) = \max_{a \in A} \left(\sum_{s \in S_a} U(s) P(s \mid E, a) \right),$$

$$EU(\alpha_f \mid E, F=f) = \max_{a \in A} \left(\sum_{s \in S_a} U(s) P(s \mid E, a, F=f) \right),$$

$$\alpha_f = \operatorname*{argmax}_{a \in A} EU(a \mid E, F=f).$$

8 Temporal Models

Def. 106 (Markov Property)**.** The variable $\mathbf{X}_t$ only depends on a subset $\mathbf{X}_1, \dots, \mathbf{X}_{t-1}$ ($\mathbf{X}_{0:t-1}$). The nth-order *Markov Property* is given when $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-n:t-1})$.

Def. 107 (Markov Process)**.** A sequence of random variables with the Markov Property. The variables can be divided into "state variables" X_t and "evidence variables" E_t.

Given the initial prior probability $\mathbf{P}(\mathbf{X}_0)$, the full joint probability distribution can be computed as

$$P(X_{0:t}, E_{1:t}) = P(X_0) \prod_{i=0}^{t} P(X_i \mid X_{i-1}) P(E_i \mid X_i).$$

Def. 108 (Transition Model)**.** A transition model of a Markov Process is given by $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$. If $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-1})$ is the same for all t, the process is said to be "stationary", making the model finite in size.

Def. 109 (Sensor Model)**.** A sensor model of a Markov Process predicts the influence of percepts on the belief state. It the "sensor Markov property" iff $\mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_{0:t}, E_{1:t-1}) = \mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$

Def. 110 (Filtering)**.** Computing the belief state from prior experience by dividing up the evidence (1), using Bayes' rule (2) and applying the sensor Markov property (3),

$$\begin{aligned} \mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t-1}, \mathbf{e}_t) && (1) \\ &= \alpha \mathbf{P}(\mathbf{e}_t \mid \mathbf{X}_t, \mathbf{e}_{1:t-1}) \mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t-1}) && (2) \\ &= \alpha \underbrace{\mathbf{P}(\mathbf{e}_t \mid \mathbf{X}_t)}_{\text{transit. model}} \underbrace{\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t-1})}_{\text{recursion}} && (3) \end{aligned}$$

Def. 111 (Prediction)**.** Computing a future state distribution, ie. filtering without new evidence:

$$\mathbf{P}(\mathbf{X}_{t+k+1} \mid \mathbf{e}_{1:t}) = \sum_{\mathbf{x}_{t+k}} \mathbf{P}(\mathbf{X}_{t+k+1} \mid \mathbf{x}_{t+k}) \underbrace{\mathbf{P}(\mathbf{x}_{t+k} \mid \mathbf{e}_{1:t})}_{\text{recursion}},$$

where $0 < k$.

Def. 112 (Smoothing)**.** Improving a past belief state, using Bayes' rule (1) and cond. independence (2),

$$\begin{aligned} \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t}) &= \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}, e_{k+1:t}) \\ &= \alpha \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}) \mathbf{P}(\mathbf{X}_{k+1} \mid X_k, \mathbf{e}_{1:k}) && (1) \\ &= \alpha \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}) \underbrace{\mathbf{P}(\mathbf{X}_{k+1} \mid X_k)}_{\text{recursion}} && (2) \end{aligned}$$

where $0 \le k < t$.

Def. 113 (Most likely Explanation)**.** Used to explain the what is the most probable sequence of events, that caused the perceived evidence $\max_{\mathbf{x}_1, \dots, \mathbf{x}_n} \mathbf{P}(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1})$, by calculating:

$$\mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \max_{\mathbf{x}_t} \left(\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t) \max_{\mathbf{x}_1, \dots, \mathbf{x}_n} P(\mathbf{x}_1, \dots, \mathbf{x}_{t-1}, \mathbf{x}_t \mid \mathbf{e}_{1:t}) \right).$$

Def. 114 (Hidden Markov Model)**.** A Markov model with a single state variable $X_t \in \{1, \dots, S\}$ and a single evidence variable.

Using a transition matrix $T_{ij} = P(X_t = j \mid X_{t-1} = i)$ and $O_{tii} = P(e_t \mid X_t = i)$ one can reinterpret Markov inference:

HMM filtering equation $\mathbf{f}_{1:t+1} = \alpha(\mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t})$

HMM smoothing equation $\mathbf{b}_{k+1:t} = \mathbf{T}\mathbf{O}_{k+1} \mathbf{b}_{k+2:t}$

Def. 115 (Dynamic Bayesian Network)**.** A Bayesian network with random variables indexed by a time structure, and can therefore be seen as a network with an infinite number of variables.

9 Making Complex Decisions Rationally

Def. 116 (Sequential Decisiton Problem)**.** The agent's utility depends on a sequence of decisions, incorporating utilities, uncertainty and sensing.

Def. 117 (Markov Decision Problem)**.** A sequential decision problem consisting of a set S of states, A of actions, a transition model $P(s' \mid s, a)$ and a reward function $R : S \mapsto \mathbb{R}$, in a fully observable, stochastic environment. The goal is to find an optimal policy $\pi : S \mapsto A$, mapping every state to the best action.

Def. 118 (Stationary Preferences)**.** Preferences are called "stationary" iff

$$[r, r_0, r_1, \dots] \succ [r, r_0', r_1', \dots] \iff [r_0, r_1, \dots] \succ [r_0', r_1', \dots]$$

The only ways to combine these over time is

additive $U([s_0, s_1, \dots]) = \sum_i R(s_i)$

discounted $U([s_0, s_1, \dots]) = \sum_i \gamma^i R(s_i)$, where γ is called "discount factor".

Def. 119 (Utility of States)**.** The optimal policy $\pi^* = \pi_s^* = \max_\pi U^\pi(s)$ where for a given policy π and s_t being the state an agent reaches at time t

$$U^\pi(s) := E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \right],$$

the utility $U(s)$ of a state s is $U^{\pi^*}(s)$.

Def. 120 (Bellman Equation)**.** The utilities of states is given by the solution to the equation

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \left(\sum_{s'} U(s') P(s' \mid s, a) \right)$$

Def. 121 (Value Iteration Algorithm)**.** A method to find the fix-point of the Bellman equation:

Algo. 1 ValueIteration(mdp, ϵ) **returns** a utility fn.

Input mdp a MDP (S, $A(s)$, $P(s' \mid s, a)$, $R(s)$, γ), the maximum permitted error ϵ

```
repeat
    U := U', δ := 0
    for each state s in S do
        U'[s] := R(s) + γ max_a Σ_{s'} U[s'] P(s' | s, a)
        δ := max{|U'[s] − U[s]|, δ}
until δ < ε(1 − γ)/γ
```

Def. 122 (Policy Iteration Algorithm)**.** Algorithm for iteratively evaluating and improving policies until no changes are made:

Algo. 2 PolicyIteration(mdp) **returns** a policy

Input mdp a MDP (S, $A(s)$, $P(s'\,|\,s,a)$, $R(s)$, γ)

repeat
  $U :=$ PolicyEvaluation(π,U,mdp)
  unchanged := true
  for each state s in S **do**
    $a^* := \text{argmax}_{a\in A(s)}(\sum_{s'} P(s'\,|\,s,a)U(s'))$
    best $:= \sum_{s'} P(s'\,|\,s,a^*)U(s')$
    if best $> \sum_{s'} P(s'\,|\,s,\pi[s'])U(s')$ **then**
      $\pi[s] := a^*$,unchanged $:= false$
until unchanged ▷ U satisfies Bellman equation

Def. 123 (Partially Observable MDP)**.** A MDP with a sensor model O that is stationary ($O(s,e) = P(e\,|\,s)$), i.e. the agent does not know in what state it is. To update the belief state the agent calculates

$$b'(s') = \alpha P(e\,|\,s') \sum_s P(s'\,|\,s,a) b(s)$$

where a is the action taken. An agent searches through the belief state by executing the best assumed action, and updating the belief state based on the percept e.

Def. 124 (Dynamic Decision Network)**.** The extension of a DBN by action and utility nodes.

10 Machine Learning

Def. 125 (Inductive Learning)**.** Learning by examples of the form (x,y), where x is an "input sample and y a "classification". The set of examples S is called consistent if a function.

An inductive learning problem $\mathcal{P} = \langle \mathcal{H}, f\rangle$ attempts to find a hypothesis $h \in \mathcal{H}$ for a consistent training set f ($f \simeq h|_{\mathbf{dom}(f)}$).

Def. 126 (Decision Tree)**.** A tree that given examples described by attribute ("attribute-based representation"), labels non-leaf nodes with attribute-choices and leafs with classifications.

Having more "significant" attributes closer to the root helps generate a compact tree. The act of trying to find a smaller tree is called "decision tree learning".

Def. 127 (Entropy of the Prior)**.** The information of an answer to the prior probabilities $\langle p_1,\dots,p_n\rangle$ is

$$I(\langle p_1,\dots,p_n\rangle) = -\sum_{i=1}^{n} p_i \log_2(p_i).$$

Def. 128 (Information Gain)**.** Said for testing an attribute A,

$$\text{Gain}(A) = I(\mathbf{P}(C)) - \sum_a P(A=a) I(\mathbf{P}(C\,|\,A=a)),$$

and given previous results $B_1 = b_1,\dots,B_n = b_n$ is

$$\text{Gain}(A\,|\,b) = I(\mathbf{P}(C\,|\,b)) - \sum_a P(A=a\,|\,b) I(\mathbf{P}(C\,|\,a,b)),$$

where C is a classification with an estimate of the probability distribution, e.g.

$$\mathbf{P}(C) = \left\langle \frac{p}{p+n}, \frac{n}{p+n} \right\rangle,$$

for p positive and n negative examples.

Def. 129 (Learning Curve)**.** Percentage of correct test results as a function of the training set size. May encounter difficulties, if the to be approximated function is not in the hypothesis space.

Def. 130 (Overfitting)**.** When a hypothesis h assumes an error is significant part of the underlying data. Conversely, "underfitting" occurs when h cannot capture the underlying trend of the data.

Def. 131 (Decision Tree Pruning)**.** For learned decision trees: Repetitively finding test nodes and replacing it with leaf nodes if it has a low information gain, as determined by a statistical significance test.

Def. 132 (Generalization)**.** Given a set of examples $\mathcal{E}$ and a prior probability $\mathbf{P}(X,Y)$ the *Generalized Loss* is

$$\text{GenLoss}_L(h) := \sum_{(x,y)\in\mathcal{E}} L(y,h(x)) P(x,y),$$

where L is a loss function, quantifying the lost utility by a hypothesis h such as

absolute value $L_1(y,\hat{y}) = |y-\hat{y}|$
squared error $L_2(y,\hat{y}) = (y-\hat{y})^2$
0/1 $L_{0/1}(y,\hat{y}) = 0$ if $y = \hat{y}$ otherwise, 1

Def. 133 (PAC learning)**.** Any algorithm that returns a probably approximatly correct hypothesis, ie. that after a sufficiently large training set, it is unlikely to be seriously wrong. The "error rate" function

$$\text{error}(h) := \text{GenLoss}_{L_{0/1}}(h)$$

describes the probability that h will misclassifying a new example.

Def. 134 (Decision List)**.** A sequence of literal conjunctions each specifying the value to be returned if satisfied, or continuing on to the next test otherwise.

Def. 135 (Classification and Regression)**.** An inductive learning problem $\langle\mathcal{H}, f\rangle$ is a *classification* problem, iff **codom**(f) is discrete and *regression* otherwise.

Def. 136 (Linear Regression)**.** Given a weight vector $\mathbf{w} = (w_0, w_1)$ and $h_w(x) = w_1 x + w_0$, the task of finding the best $\mathbf{w}$ for a set of examples is called *linear regression*. The space of weight combinations is called the *weight space*.

Def. 137 (Gradient Descent)**.** An algorithm for finding the minimum of a continuous function f by hill climbing in the direction of steepest descent. For each vector component the calculation

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} f(\mathbf{w})$$

until $\mathbf{w}$ converges, where α is called the "learning rate".

Def. 138 (Perceptron Learning Rule)**.** A learning rule given an example $(\mathbf{x},y)$ updating

$$w_i \leftarrow w_i + \alpha(y - h_{\mathbf{w}}(\mathbf{x}))x_i$$

Def. 139 (Logistic Regression)**.** Using a "softer" learning rule, linear regression can be replaced by using a logistic function

$$h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{wx}}}.$$

10.1 Artificial Neuronal Networks

Def. 140 (Neuronal Network)**.** A directed graph, propagating *activations* a_i from *unit* i to *unit* j via *links* with *weights* $w_{i,j}$.

If the network has cycles, it is called "recurrent", otherwise "feed-forward" and it is said to have layers $\{L_0,\dots,L_n\}$

Def. 141 (McCulloc-Pitts unit)**.** A unit model where each activation is computed by

$$a_i = g\left(\sum_j w_{j,i} a_j\right),$$

given a activation function g. If g is a threshold function (e.g. logistic function) the unit is called a "perceptron unit".

Def. 142 (Perceptron Network)**.** A feed-forward network of perceptron units. Units not part of the input or output layer are called "hidden".

Def. 143 (Backpropagation)**.** Learn in a perceptron network is implemented by updating weights

$$w_{k,j} \leftarrow w_{k,j} + \alpha a_k \Delta_j$$

where

$$\Delta_j \leftarrow g'\left(\sum_j w_{j,i} a_j\right) \sum_i w_{j,i} \Delta_i.$$

Def. 144 (Backpropagation Algorithm)**.** Given a network and a set of examples, the algorithm propagates the example input through the network, then propagates deltas backwards towards the input layer and then updates every weight using the calculated delta values.

10.2 Statistical Learning

Def. 145 (Bayesian Learning)**.** Calculating the probabilities of each hypothesis, and acting upon these predictions

$$P(\mathbf{d} \mid h_i) = \alpha P(\mathbf{d} \mid h_i) P(h_i),$$

where $P(\mathbf{d} \mid h_i)$ is the likelihood to observe data $\mathbf{d} \in \mathbf{D}$ given a hypothesis, and $P(h_i)$ is the "hypothesis prior".

To predict a unknown quantity X, one uses

$$P(X \mid \mathbf{d}) = \sum_i \mathbf{P}(X \mid h_i) P(h_i \mid \mathbf{d}).$$

Def. 146 (Naive Bayes' Models for Learning)**.** Using a naive Bayes' model, a single "class" variable C is predicted given "attribute" variables X_i:

$$\mathbf{P}(C \mid X_1, \ldots, X_n) = \alpha \mathbf{P}(C) \prod_i \mathbf{P}(x_i \mid C),$$

whereafter the most likely class is chosen.

11 Natural Language Processing

Def. 147 (Natural Language Processing)**.** The intersection between computer science, artificial intelligence and linguistics, attempting to understand and generate natural language.

Def. 148 (Linguistically Realized)**.** When a piece of information i can be traced back to a fragment of an utterance U.

Def. 149 (Language Model)**.** A probability distribution of a sequence of characters or words.

Def. 150 (Text Corpus)**.** A large, structured set of texts, used for statistical analysis.

Def. 151 (n-gram model)**.** A $n-1$'th order Markov chain, that generates a character/word sequence (n-gram) of the length n. The probability of a character sequence $\mathbf{c}_{1:N}$ is

$$P(\mathbf{c}_{1:N}) = \prod_{i=1}^{N} P(c_i \mid \mathbf{c}_{1:i-1}).$$

This can be used for genre classification, named entity recognition, language generation, among other things.

For language identification, the most probable language ℓ^* of a text corpus can be approximated by applying Bayes' rule (1) and the Markov property (2):

$$\begin{aligned}\ell^* &= \operatorname{argmax}_\ell (P(\ell \mid \mathbf{c}_{1:N})) \\ &= \operatorname{argmax}_\ell (P(\ell) P(\mathbf{c}_{1:N} \mid \ell)) \qquad (1) \\ &= \operatorname{argmax}_\ell \left(P(\ell) \prod_{i=1}^{N} P(c_i \mid \mathbf{c}_{i-n:i-1}) \right), \qquad (2)\end{aligned}$$

given the, if necessary estimated, prior probabilities for $P(\ell)$.

Def. 152 (Term Frequency)**.** The number of times a word t occurs in a document d ($\mathrm{tf}(t,d)$).

Def. 153 (Inverse document frequency)**.** Calculated for a document collection $D = \{d_1, \ldots, d_n\}$:

$$\mathrm{idf}(t,D) = \log_{10}\left(\frac{n}{|\{d \in D \mid t \in d\}|}\right)$$

Def. 154 (Term Frequency/Inferse Document Frequency)**.** Combining term frequency and inverse document frequency into

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d)\,\mathrm{idf}(t,D).$$

Def. 155 (Word Embeddings)**.** A mapping of words into a $\mathbb{R}^n$ vector space. For if-idf it is given by

$$e : t \mapsto \langle \mathrm{tfidf}(t,d_1,D), \ldots, \mathrm{tfidf}(t,d_{\#(D)},D) \rangle.$$

Def. 156 (Cosine Similarity)**.** Calculated for two vectors A and B, and the angle θ between them,

$$\cos\theta = \frac{A \cdot B}{\|A\|_2 \|B\|_2}$$

holds. Used by word embeddings for information retrieval.

Def. 157 (Grammar)**.** A tuple $\langle N, \Sigma, P, S \rangle$ where
N is a finite set of non-terminal symbols,
Σ is a finite set of terminal symbols,
P is a set of production rules,
S is a distinguished "start symbol"
These can be categorized as "context-sensitive", "context-free", "regular", etc.

Def. 158 ((Formal) Language)**.** A set of sentences that can be generated by a grammar, written $L(G)$.

Def. 159 (Ambiguity)**.** Real languages pose issues for natural language processing, such as *ambiguity*, *anaphora*, *indexicality*, *vagueness*, *discourse structure*, *metonymy*, *metaphor* and *noncompositionality*.