1 Information Theory

Information Content (Surprisal)
$-log_2 p(x)$: the more frequent the word, the lower its information content. e.g. the word type "blue" occurs ca. 3750 times in 10000 tokens, and its information content is $-log_2(3750/10000) \approx 1.42$ bits.

Shannon Entropy
$H(X) = -\sum p(x) log_2 p(x)$: entropy as probability, the average information content of information encoding units in the language. Measure of information encoding potential of a symbol system. The higher the uncertainty, the larger the entropy. e.g. $H_{char}(Morse) = -(\frac{86}{136} * log_2(\frac{86}{136}) + \frac{50}{136} * log_2(\frac{50}{136})) \approx 0.949$ bits per character.

Joint Entropy, Conditional Entropy
Joint Entropy: $H(X,Y) = -\sum\sum p(x,y) log_2 p(x,y)$
Conditional Entropy: $H(Y|X) = -\sum p(x) \sum p(y|x) log_2 p(y|x)$ The more ambiguity in language (uncertainty), the higher conditional entropy. No ambiguity -> 0.

Probability Estimation
Maximum Likelihood (ML)
Problems: unit problem, sample size problem, interdependence problem, extrapolation problem
Methods: frequency-based, language models, experiments with humans

Mutual Information
$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$
reduction in the uncertainty of X given Y. compromise between minimum learning cost H(Y) and maximum expliciteness I(X; Y).
Entropy is the upper bound on the mutual information between forms and meanings
Is the entropy rate zero? -> asymptotic determinism of human utterances.

2 Propositional Logic

Why formal logic? overcome ambiguity, determine relationships between meanings of sentences, determine meanings of setences, model compositionality, recursive system.

Definition
Proposition: The meaning of a simple declarative sentence
Extensions: real-world situations they refer to.
Frege's Generalization: The extension of a sentence S is its truth value
The proposition expressed by a sentence is the set of possible cases [situations] of which that sentence is true.
propositional variables: p, q, r... propositional operators: $\neg$, $conjunction \wedge$, $disjunction \vee$, XOR, $\rightarrow$, $\leftrightarrow$

Syntax: Recursive Definition

1. Propositional letters in the vocabulary of L are formulas in L.
2. If ϕ is a formula in L, then $\neg\phi$ is too.
3. If ϕ and ψ are formulas in L, then $(\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$ are too.
4. Only that which can be generated by the clauses (i)-(iii) in a finite number of steps is a formula in L.

invalid: $\neg(\neg\neg p), \neg((p \wedge q))$

Construction Trees

- $(\neg(p \vee q) \rightarrow \neg\neg q) \leftrightarrow r$ $(iii. \leftrightarrow)$
  - $(\neg(p \vee q) \rightarrow \neg\neg q)$ $(iii. \rightarrow)$
    - $\neg(p \vee q)$ (ii)
      - $p \vee q$ $(iii. \vee)$
        - p (i)
        - q (i)
    - $\neg\neg q$ (ii)
      - $\neg q$ (ii)
        - q (i)
  - $r (i)$

Valuation Functions
For every valuation V and for all formulas ϕ:
$V(\phi \leftrightarrow \psi) = 1 iff V(\phi) = V(\psi)$.

3 Predicate Logic

Definition
constant symbols: a, b, c, ...
variable symbols: x, y, z, ...
n-ary/n-place predicate symbols: A, B, C, reflect relations between n elements (n>0)
connectives: $\neg, \wedge, \vee, \rightarrow, ...$
quantifiers: $\forall, \exists$
round brackets (), equal sign =

Syntax: Recursive Definition

1. If A is an n-ary predicate letter in the vocabulary of L, and each of t1,..., tn is a constant or a variable in the vocabulary of L, then At1,..., tn is a formula in L.
2. If ϕ is a formula in L, then $\neg\phi$ is too.
3. If ϕ and ψ are formulas in L, then $(\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$ are too.
4. If ϕ is a formula in L and x is a variable, then $\forall x\phi$ and $\exists x\phi$ is too.
5. Only that which can be generated by the clauses (i)-(iv) in a finite number of steps is a formula in L.

invalid: $a, A, \forall(Axy)$

Construction Trees

- $\forall_x \forall_y ((A_{xy} \wedge B_y) \rightarrow \exists_x A_{xb})$ $(iv.\forall)$
  - $\forall_y ((A_{xy} \wedge B_y) \rightarrow \exists_x A_{xb})$ $(iv.\forall)$
    - $(A_{xy} \wedge B_y) \rightarrow \exists_x A_{xb}$ $(iii. \rightarrow)$
      - $A_{xy} \wedge B_y$ $(iii.\wedge)$
        - A_{xy} (i)
        - B_y (i)
      - $\exists_x A_{xb}$ $(iv.\exists)$
        - A_{xb} (i)

Semantics
domain (D): set of entities
interpretation functions I = $\{< m, e >, < s, e >, < v, e >\}$, I(m) = e, I(s) = e, I(v) = e.
model M: consists of a domain D and an interpretation function I which conforms to:

1. if c is a constant in L, then I(c) ∈ D
2. if B is an n-ary prpedicate letter in L, then I(B) ⊂ D

valuation function V_M:
If Aa1,...,an is an atomic sentence in L, then V_M(Aa1,...,an) = 1 iff <I(a1), ..., I(an)> ∈ I(A).
...
$V_M(\forall x\phi) = 1$ iff $V_M([c/x]\phi) = 1$ for all constants c in L.
$V_M(\exists x\phi) = 1$ iff $V_M([c/x]\phi) = 1$ for at least one constant c in L.
If $V_M(\phi) = 1$, then ϕ is said to be true in model M.

Formula vs. Sentence
A sentence is a formula in L which lacks free variables.
Sentence: Aa, $\forall x(Fx)$, $\forall x(Ax \rightarrow \exists y By)$
Not a sentence (but Formula): Ax, Fx, $Ax \rightarrow \exists y By$

4 Second-Order Logic

e.g. CR (CX: X is a predicate with the property of being a color; Rx: x is red)
$\exists X(CX \wedge Xm)$: Mars has a color.
$\exists X(Xj \wedge Xp)$: John has at least one thing in common with Peter.
$\forall x(\exists X((AX \wedge Xx) \wedge Jx) \rightarrow \exists Y(Yx \wedge CY))$: All animals that live in the jungle have a color.

Vocalbulary extention
First-order predicate variables: X, Y, Z, ... Second-order predicate constants: A, B, C, ... e.g. AX: X is a property with the property of being an animal

Syntax: Recursive Definition

1. If A is an n-ary first-order predicate letter in the vocabulary of L, and each of t1,..., tn are individual terms in L, then At1,..., tn is an (atomic) formula in L.
2. If X is a [first-order] predicate variable and t is an individual term (both constants and variables) in L, then Xt is an atomic formula in L;
3. If A is an n-ary second-order predicate letter/constant in L, and T1,...,Tn are first-order **unary** predicate constants, or predicate variables, in L, then AT1,...,Tn is an (atomic) formula in L;
4. If ϕ is a formula in L, then $\neg\phi$ is too.
5. If ϕ and ψ are formulas in L, then $(\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$ are too.
6. If ϕ is a formula in L and x is a variable, then $\forall x\phi$ and $\exists x\phi$ is too.
7. If X is a [first-order] predicate variable, and ϕ is a formula in L, then $\forall X\phi$ and $\exists X\phi$ is too.
8. Only that which can be generated by the clauses (i)-(vii) in a finite number of steps is a formula in L.

invalid: $x, X, Xab, \forall(Xa)$

Semantics
just as a first-order predicate denotes a set of entities, a second-order predicate denotes a set of a set of entities.

5 Type theory

Tools to get to grips with frequent compositional structures in natural language (adj-n, adv-v, art-n, prep-np... combis)

Definition

1. $e, t \in T$
2. if $a, b \in T$, then $< a, b > \in T$
3. nothing is an element of T except on the basis of clauses (i) and (ii).

invalid: et, <e,e,t>, <e,<e,t>

Functional Application
If α = <e,t> and β = e then $\alpha(\beta)$ = t.
If α = <t,<t,e» and β = <t,e> then $\alpha(\beta)$ is not defined.

Semantic Types
individual: e
sentences: t
1-place predicates (intransitive verb): <e,t>
2-place predicates (transitive verb): <e,<e,t»
3-place predicates (ditransitive verb): <e.<e,<e,t»>
common nouns (e.g. dog): <e,t>
NP (e.g. the dog): e
determiners (e.g. the): <e,<e,t»
adjectives: <e,t>
adjectives as predicate modifiers(e.g. *happy* dog): «e,t>,<e,t»
adverbs(predicate modifier): «e,t>,<e,t»
sentence modifier(e.g. not): <t,t>
function(entity to entity) (e.g. the farther of): <e, e>
One-place second-order predicate: «e,t>,t>
Two-place second-order predicate: «e,t>,«e,t>,t»

Syntax: Recursive Definition

1. If α is a variable or a constant of type a in L, then α is an expression of type a in L.
2. If α is an expression of type <a,b> in L, and β is an expression of type a in L, then $(\alpha(\beta))$ is an expression of type b in L.
3. If ϕ and ψ are formulas in L, then so are $\neg\phi, (\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$.
4. If ϕ is an expression *of type t* in L and v is a variable (of arbitrary type a), then $\forall v\phi$ and $\exists v\phi$ are expression of type t in L.
5. If α and $beta$ are expressions in L which belong to the same type, then $(\alpha = \beta)$ is an expression of type t in L.
6. Every expression L is to be constructed by means of (i)-(v) in a finite number of steps.

The *formulas* are those expressions which are of type t.

Difference to Predicate Logic:
Jumbo befriends Maya.
Predicate Logic: Bjm
Type-theoretic logic: (B(m))(j) or alternatively B(m)(j)

Semantics

Truth valuations via pariticular interpretation functions defined for different types of expressions. e.g. interpretation function I for which it holds that: I(W)(d) = 1 iff d∈W, otherwise 0.