ECS763 Natural Language Processing

Unit 8: Logical and Distributional Semantics

Lecturer: Julian Hough
School of Electronic Engineering and Computer Science

OUTLINE

1) Semantics
2) First Order Logic and Semantic Models
3) Higher-order logic and mapping syntax to semantics
4) Distributional Hypothesis for meaning
5) Vector-space semantics
6) Smoothing parameters and similarity measures

OUTLINE

1) Semantics
2) First Order Logic and Semantic Models
3) Higher-order logic and mapping syntax to semantics
4) Distributional Hypothesis for meaning
5) Vector-space semantics
6) Smoothing parameters and similarity measures

Semantics

- With syntax we looked at the form of sentences of natural language, now we need to look at the **meaning** of sentences- that is the study of **semantics**.

- For our applications to become useful, we need our systems to **interpret or understand** the user's language correctly.

- Semantic information or semantic representations bridge the gap between syntax and our **knowledge about the world**.

- We need a **semantics** whenever the syntactic information is not enough to do the processing needed for the task at hand.

    - This is the case for most real-world applications, beyond parsing or language modelling. Producing the syntactic tree for 'put the remote control on the table' won't put it there without extra reasoning!
    - We want our systems to **do something for us by understanding our written and spoken behavioural input**.

Semantics

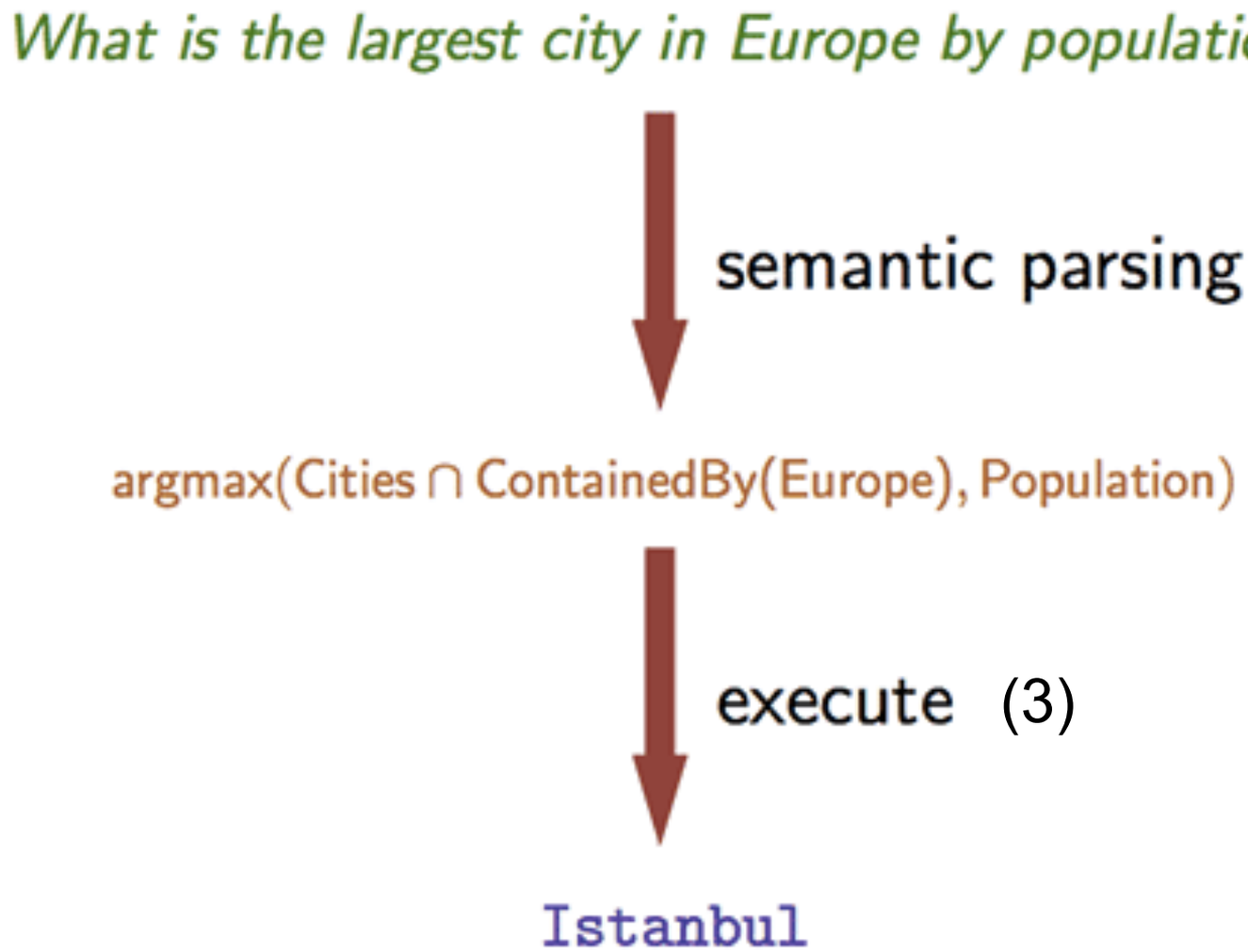
- Most human written/spoken communication needs **semantic processing**, that is, determining what sentences and words mean in the real world):

  - Following a recipe: you are given a list of linguistic expressions and are supposed to follow them with the goal of preparing some food.
  - Answering exam and quiz questions.
  - Understanding jokes and insults.
  - Learning a programming language using a book.
  - etc…

- Applications are the same, needing **natural language understanding (NLU) capabilities**, but state-of-the-art systems are more constrained:

  - Ordering Pizzas.
  - Booking reservations/flights.
  - Asking queries of a database in a natural language.
  - Return specific internet search results.
  - Robot picking objects from commands.
  - etc…

Semantics

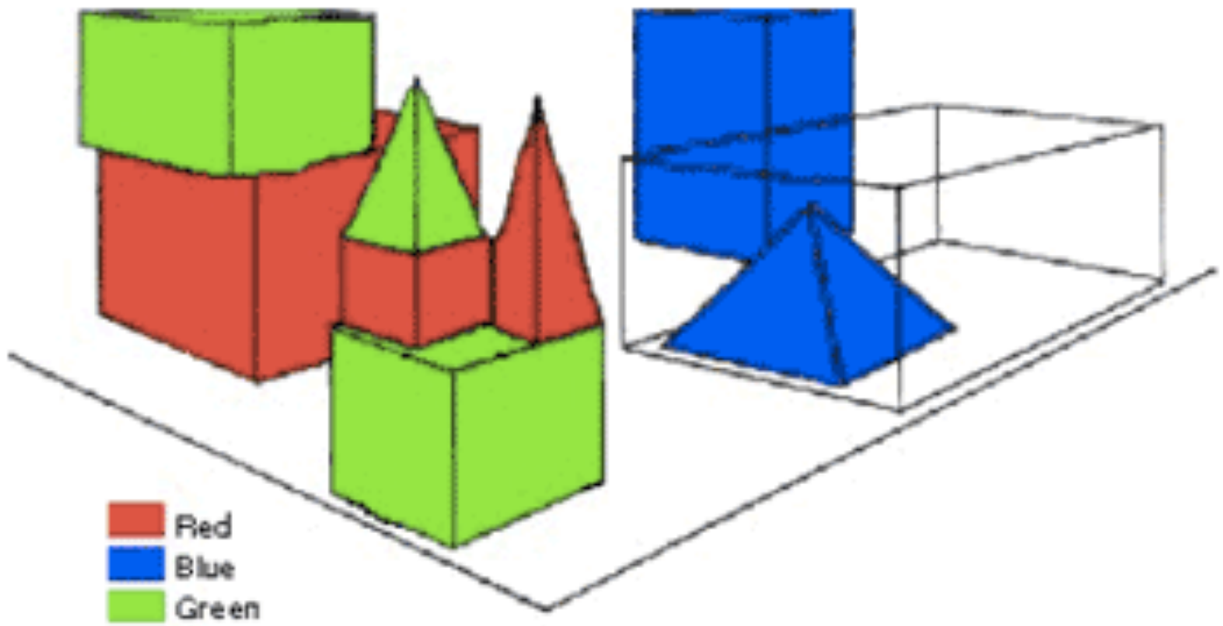


1. syntactic parse tree from words (e.g. CFG/CCG)

-> 2. **semantic formula compilation from tree**

-> 3. **Evaluation of Semantic formula from a model (ending in action)**

Semantics



Person: Pick up a big red block.

Semantic parsing (1+2)

$$\exists x\,.\,Take(x) \wedge red(x) \wedge big(x) \wedge block(x)$$

Execute (3)

```
MoveTo(object3)
Take(object3)
```

1. syntactic parse tree from words (e.g. CFG/CCG)

-> 2. **semantic formula compilation from tree**

-> 3. **Evaluation of semantic formula from a model (ending in action)**

Semantics

- There are different kinds of semantic representations which can apply to a number of target domains/situations and reasoning can be done with them for achieving different tasks:

  - **First Order Logic**
  - **Abstract Meaning Representations**
  - **Frames**

Semantics

- There are different kinds of semantic representations around:
- **First Order Logic** semantic representation for the sentence “I have a car”

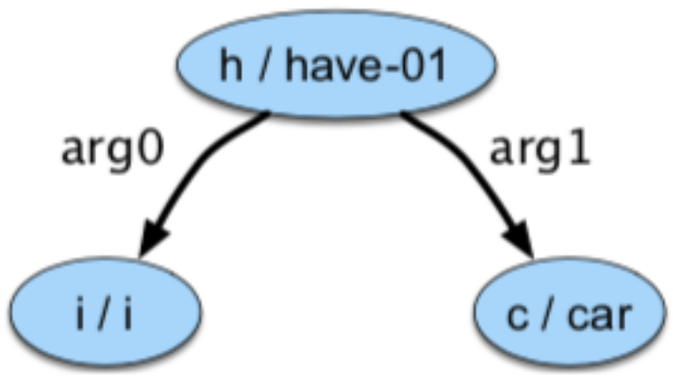
$$\exists x, yHaving(x) \wedge Haver(Speaker, x) \wedge HadThing(y, x) \wedge Car(y)$$

Note here the *x* is some event-like entity (i.e. the having event)
The *y* is the entity that is a car (i.e. the car).

Semantics

- There are different kinds of semantic representations around:
- **Abstract Meaning Representation** for the sentence “I have a car”

```
(h / have-01
        arg0: (i / i)
        arg1: (c / car))
```



Semantics

- There are different kinds of semantic representations around:
- **Frame or record** representation for “I have a car”

```
Having
 Haver:  Speaker
 HadThing:  Car
```

Semantics

- We are moving from a natural language:
  - the language of English

- to another formal language:
  - a semantic representation language

- The semantic representation language cannot be just any set of symbols. It has to satisfy a set of properties:

  - 1- Verifiability
  - 2- Unambiguous Representations
  - 3- Canonical Forms
  - 4- Expressiveness

Verifiability

- We should be able to use the representation to be able to **verify**/ check whether what the syntax is expressing holds (is true) in the world, as we know it.

Verifiability

- syntax: expression (a) ``Kipling serves vegetarian food”
- semantics: expression (b) Serves(Kipling, Vegetarian(food))

- Can we use (b) to tell if (a) is true? Can we use (b) to verify (a)?

- If we had a Knowledge base, with properties “food”, “Vegetarian”, and “serves” we could indeed do so.

- We could go and query the knowledge base using (b) and see whether it holds or not.

- We are indeed assuming that the knowledge base is representing the information of the world and correctly so.

Unambiguous Representations

- I want to eat somewhere close by.

- This can mean two things:
  - 1- I want to eat something at a place that is close to me.
  - 2- I want to eat a place that is close to me.

- A good semantic representation should be able to represent both.

- While the Natural Language query can be ambiguous, our meaning representation itself **cannot be ambiguous.**

Unambiguous Representations

- I want to eat delicious food.
- Here, it is not very clear cut what delicious food means.
- It is a vague concept that varies from person to person and culture to culture.
- Other similar concepts are adjectives such as ``tall, big, small".
- These kinds of considerations fell under semantic vagueness. Our semantic representations should be able to deal with vague concepts.

Canonical Form

- Kiplings serves vegetarian food.
- Vegetarian food is served at Kiplings.
- Kiplings has vegetarian food.
- Kiplings serves vegetarian meals.
- Kiplings serves vegetarian dishes.

- All of these syntactic forms mean the same things. Their semantic representations should also be the same.

- In other words, if one of them is true in the world, so should be others.

- Or, when two distinct inputs that mean the same thing should have the **same meaning representation**

Expressiveness

- Finally, a meaning representation scheme must be **expressive enough** to handle a wide range of subject matter, ideally any sensible natural language utterance.

- Although this is probably too much to expect from any single representational system, First-Order Logic is expressive enough to handle quite a lot of what needs to be represented for NLP.

OUTLINE

1) Semantics
2) First Order Logic and Semantic Models
3) Higher-order logic and mapping syntax to semantics
4) Distributional Hypothesis for meaning
5) Vector-space semantics
6) Smoothing parameters and similarity measures

First Order Logic Semantics

- We should be able to use the representation to be able to tell whether what the syntax is expressing holds (is true) in the world, as we know it.

- syntax: expression (a) ``Kipling serves vegetarian food”
- semantics: expression (b) Serves(Kipling, Vegetarian(food))

- Can we use (b) to tell if (a) is true? Can we use (b) to verify (a)?

FOL Semantic Representation Language

We are defining a representation language. This constitutes of:

(I) Two sets of symbols that make the vocabulary of this language:
    (1) Logical symbols: /\, \/, =>, quantifiers
    (2) Non-Logical symbols: John, Mary, eat, vegetarian, food

(II) A set of rules that tells us how to mix (1) and (2) to get sentences.

(III) Each of (2) has a *denotation* (meaning) in a model.

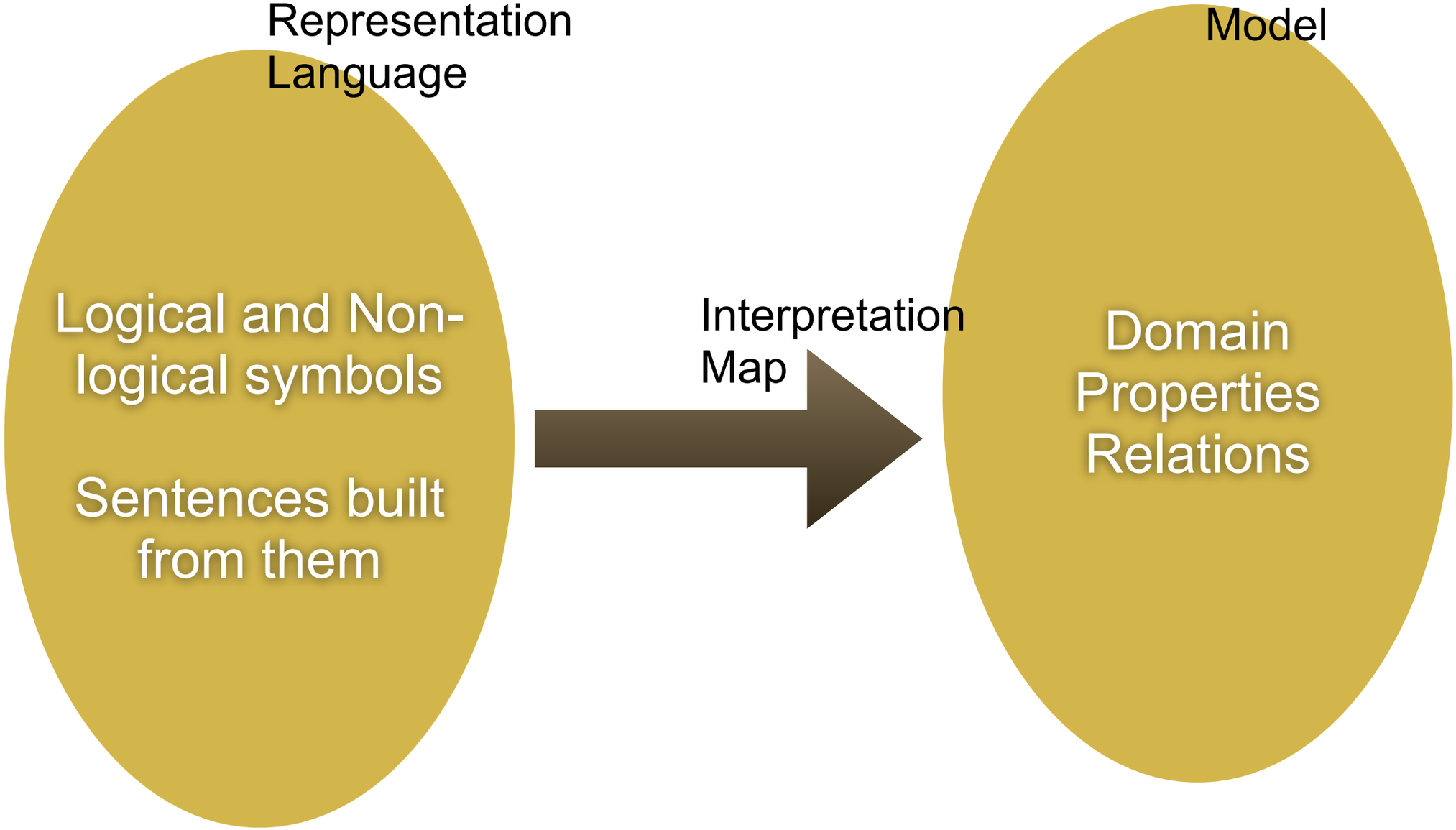
(IV) An interpretation map between 2 and (III): between every non-logical symbol and its denotation. This map is compositionally extendible to sentences using (II).

FOL Semantic Representation Language

A model consists of the following elements:

(I) Domain: a set of individuals/symbols
(II) Properties: sets of individuals
(III) Relations: sets of tuples of individuals

FOL Semantic Representation Language



Example

Representation Language

and is our only logical symbol.

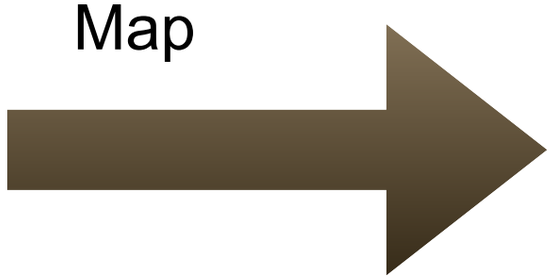
All the other symbols, i,e.
John, likes, vegetarian food, Mary, steak are non-logical.

Example

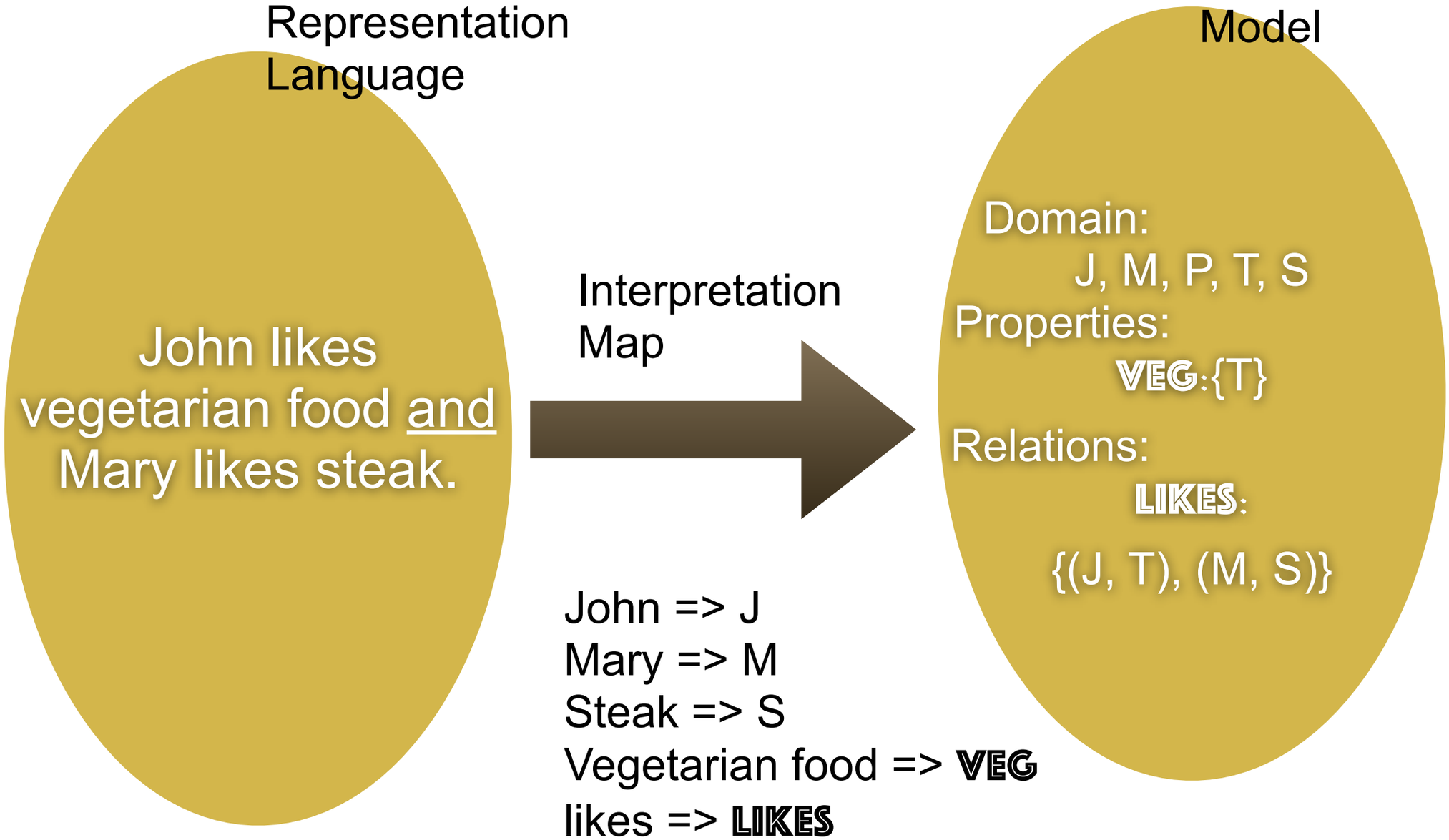
Representation Language

John likes vegetarian food and Mary likes steak.

Interpretation Map



Example


Representation Language
John likes vegetarian food and Mary likes steak.
Interpretation Map
Model
Domain:
J, M, P, T, S
Properties:
VEG:{T}
Relations:
LIKES:
{(J, T), (M, S)}
John => J
Mary => M
Steak => S
Vegetarian food => VEG
likes => LIKES

FOL Semantic Representation Language

Is our meaning representation verifiable? If so, how?

According to the interpretation map, verifying whether

“John likes vegetarian food and Mary likes Steak”

is true, we have to check whether
VEG(T) /\ LIKES(J, T) /\ LIKES (M, S)

holds in our model.

FOL Semantic Representation Language

- Is our meaning representation verifiable? If so, how?
- According to the interpretation map, verifying whether "John likes vegetarian food and Mary likes Steak" is true, we have to check whether the following expression holds in our model:
  - VEG(T) /\ LIKES(J, T) /\ LIKES (M, S)

- That is, we have to check whether the following hold:
  - T has/belongs to the property VEG
  - J is related to T via the relation LIKES
  - M is related to S via the relation LIKES

- These are all true, so our sentence is true.

FOL Semantic Representation Language

- We can do more!
- We can also verify that the following sentences are true in our model:
  - “Mary does not like tofu.”
  - “John does not like steak.”
  - “Not every one likes tofu”
  - “Not every one likes tofu or steak”
  - “Not every one likes vegetarian food”
  - “Either John or Mary does not like vegetarian food”

FOL Semantic Representation Language

- We can do more!
- We can also verify that the following sentences are true in our model:
  - "Mary does not like tofu." $\neg like(M, T)$
  - "John does not like steak." $\neg like(J, S)$
  - "Not every one likes tofu" $\neg \forall x \,.\, like(x, T)$ equiv to $\exists x \,.\, \neg like(x, T)$
  - "Not every one likes tofu or steak"

    $\neg \forall x \,.\, like(x, T) \vee like(x, S) =$ $\exists x \,.\, \neg like(x, T) \wedge \neg like(x, S)$
  - "Not every one likes vegetarian food"

    $\neg \forall x \exists y \,.\, likes(x, y) \wedge veg(y)$
  - "Either John or Mary does not like vegetarian food"

    $\exists y \,.\, (\neg like(J, y) \vee \neg like(M, y)) \wedge veg(y)$

FOL Semantic Representation Language

- "Anyone who loves movies hates either theatre or songs."
-

$$\forall x \,.\, love(x, movies) \rightarrow (hate(x, theatre) \vee hate(x, songs))$$

- "There is someone who loves movies who hates either theatre or songs."

$$\exists x \,.\, love(x, movies) \wedge (hate(x, theatre) \vee hate(x, songs))$$

Everyone has at least one food that they like.

$$\forall x \,.\, \exists y \,.\, food(y) \wedge like(x, y)$$

(everyone has their own food
i.e. John likes bananas,
Khalid likes oranges)

(y is a set of numbers
Which are lower than
A given x)

$$\forall x \,.\, \exists y \,.\, x > y$$

(y is - infinity)

$$\exists y \,.\, \forall x \,.\, food(y) \wedge like(x, y)$$

(there's this one food,
e.g. chocolate, that
Everyone likes)

$$\exists y \,.\, \forall x \,.\, x > y$$

Truth Theoretic Semantics

How do we do that? Using the truth theoretic semantics of FOL.
This is the language of FOL:

$$
\begin{array}{rcl}
\textit{Formula} & \rightarrow & \textit{AtomicFormula} \\
 & | & \textit{Formula Connective Formula} \\
 & | & \textit{Quantifier Variable}, \ldots \textit{Formula} \\
 & | & \neg \textit{Formula} \\
 & | & (\textit{Formula}) \\
\textit{AtomicFormula} & \rightarrow & \textit{Predicate}(\textit{Term}, \ldots) \\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \\
 & | & \textit{Constant} \\
 & | & \textit{Variable} \\
\textit{Connective} & \rightarrow & \wedge \mid \vee \mid \Longrightarrow \\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & A \mid \textit{VegetarianFood} \mid \textit{Maharani} \cdots \\
\textit{Variable} & \rightarrow & x \mid y \mid \cdots \\
\textit{Predicate} & \rightarrow & \textit{Serves} \mid \textit{Near} \mid \cdots \\
\textit{Function} & \rightarrow & \textit{LocationOf} \mid \textit{CuisineOf} \mid \cdots
\end{array}
$$

Truth Theoretic Semantics

How do we do that? Using the truth theoretic semantics of FOL.
This is the its part of its truth theoretic semantics:

not/negation of

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \implies Q$
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
True	*True*	*False*	*True*	*True*	*True*

OUTLINE

1) Semantics
2) First Order Logic and Semantic Models
3) Higher-order logic and mapping syntax to semantics
4) Distributional Hypothesis for meaning
5) Vector-space semantics
6) Smoothing parameters and similarity measures

The Lambda Notation

- The Lambda notation was introduced by Church in 1940. He extended the language of FOL with lambdas in oder to make reasoning about semantics easier. Introduces the idea of a **variable** and **functional application.**

- A lambda term consists of the Greek letter Lambda, followed by variables and predicates:

Examples

$\lambda x.P(x)$

$\lambda x.$ VEG(x)

$\lambda x.\lambda y.$ LIKES(x,y)

The Lambda Notation

- We can apply lambda terms to each other and the result will be another lambda term.

$$\lambda x.P(x)(A)$$

$$P(A)$$

Examples

$\lambda x.$ VEG(x) (J) = VEG(J)

$\lambda x.\lambda y.$ LIKES(x,y)(J) (T) = LIKES(J,T)

- Most of the time natural language semantics is written using the Lambda notation.

Events, Time, Aspects

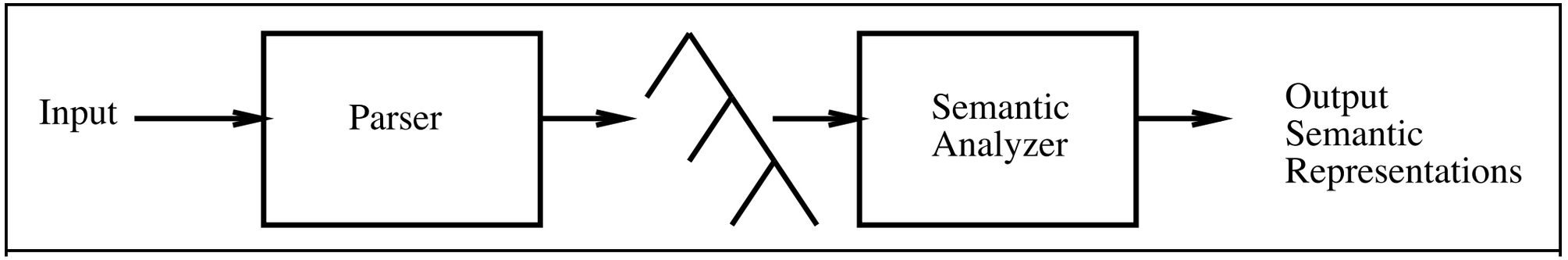
- Usually the natural language semantics can express more than just properties and relations.

- Specific properties and relations are formalised and used to reason about specific information, such as:

  - 1- Events: certain verbs express events, e.g.. eating
  - 2- Time: events happen in time, eating after or before
  - 3- Aspect: activity (eating), achievement (found), accomplishment (booked)
  - … (read the J & M book for more)

Mapping from Syntax to Semantics

- We have shown how to verify semantic representations made in FOL. But how do we get to these semantic representations from syntax?



- Equivalently, how does the above “semantic analyser" unit works?

Mapping from Syntax to Semantics

- We have learnt about different grammatical formalisms. Each gives us a different parse tree or structure. For now, let's consider the generative grammar model.
- The semantic representation of each generative parse tree is obtained by following two main steps:

- 1- To each rule of our grammar, we assign a semantic representation in the Lambda notation.
- 2- As rules are combined with each other, e.g. by working with them as rewrite rules or by going up the parse tree, the corresponding Lambda terms are "applied" to each other.

Examples

Syntactic CFG Rule | Semantics

$S \rightarrow NP\ VP$ $\quad \{VP.sem(NP.sem)\}$

- This means in order to get a semantics for the sentences, apply the semantics of VP (VP.sem) to that of NP (NP.sem).
- Similarly for the rest of the rules below:

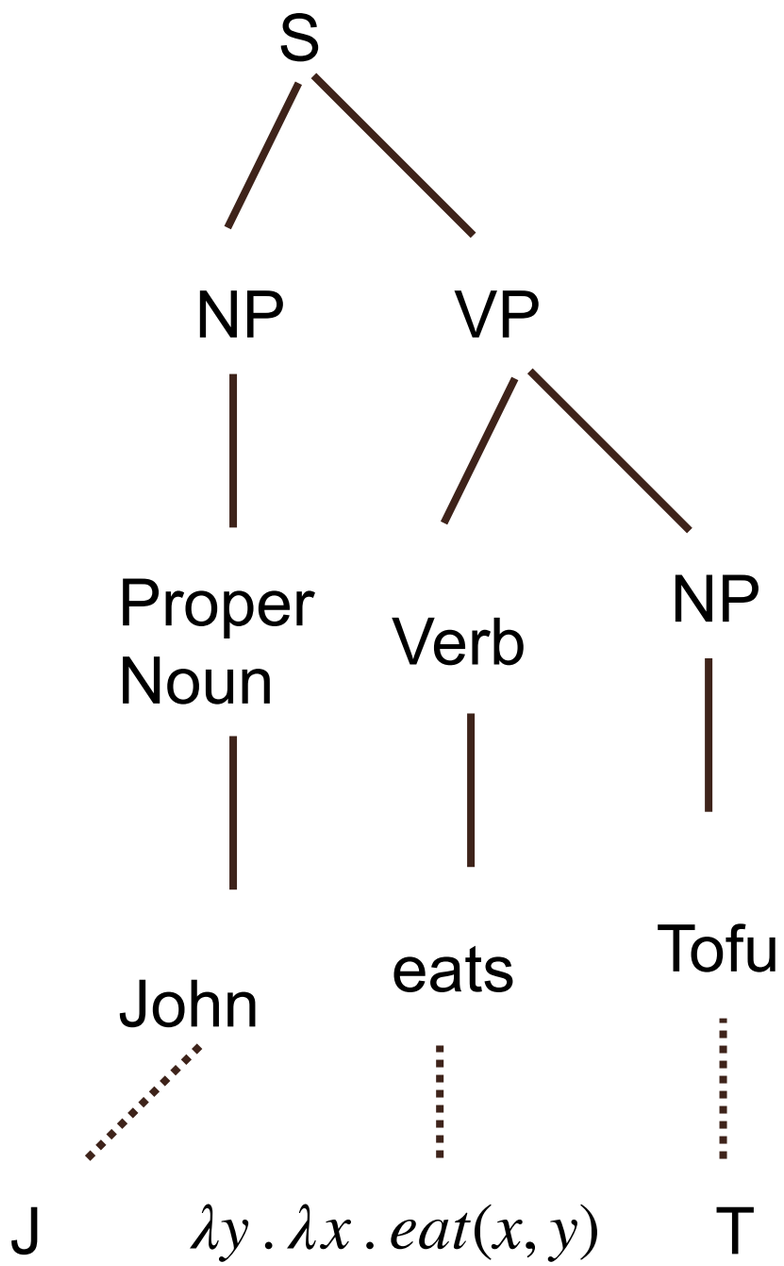
$VP \rightarrow Verb\ NP$ $\quad \{Verb.sem(NP.sem)\}$

$Verb \rightarrow serves$ $\quad \{\lambda x \exists e,y\ Isa(e,Serving) \wedge Server(e,y) \wedge Served(e,x)\}$

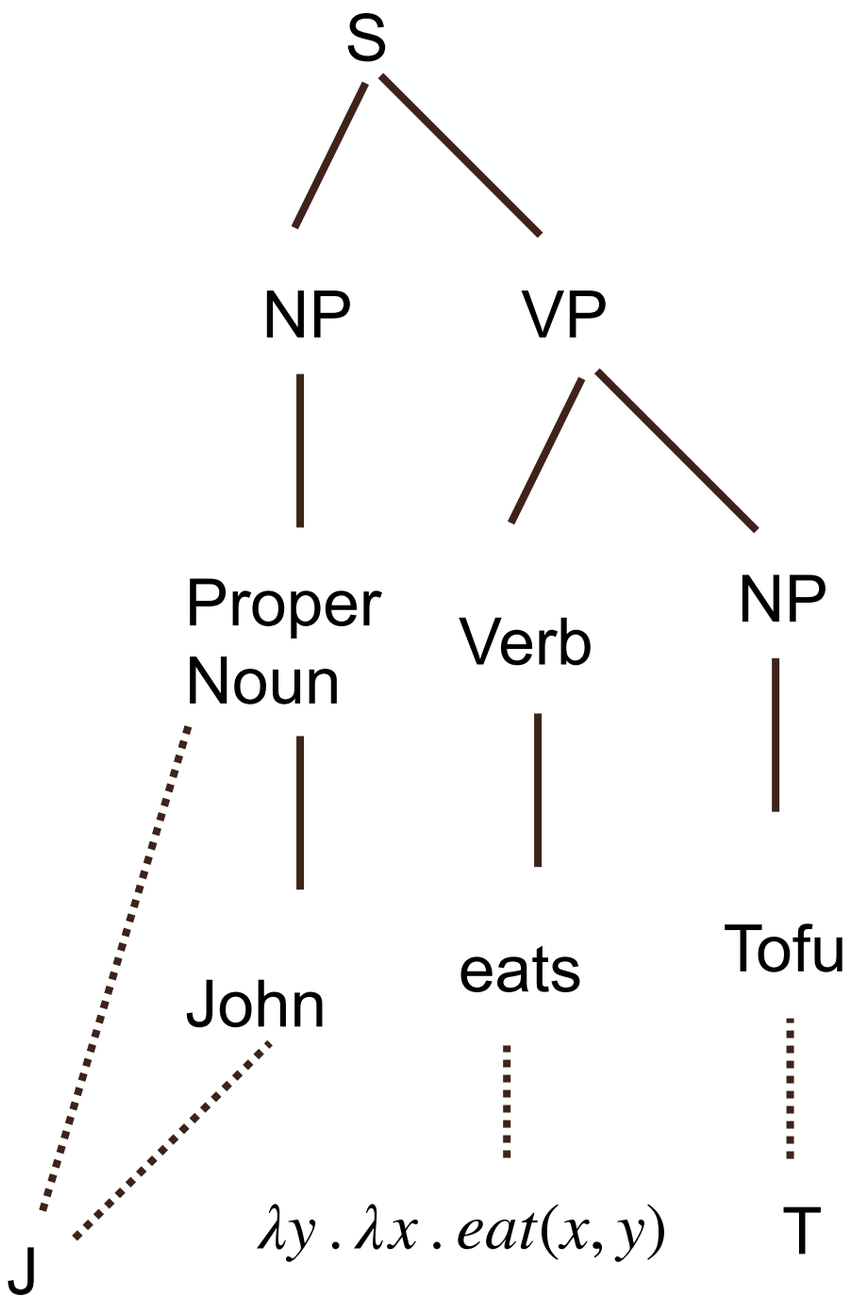
$NP \rightarrow ProperNoun$ $\quad \{ProperNoun.sem\}$

Assume a simple semantics for *eats* -> $\lambda y \,.\, \lambda x \,.\, eat(x, y)$

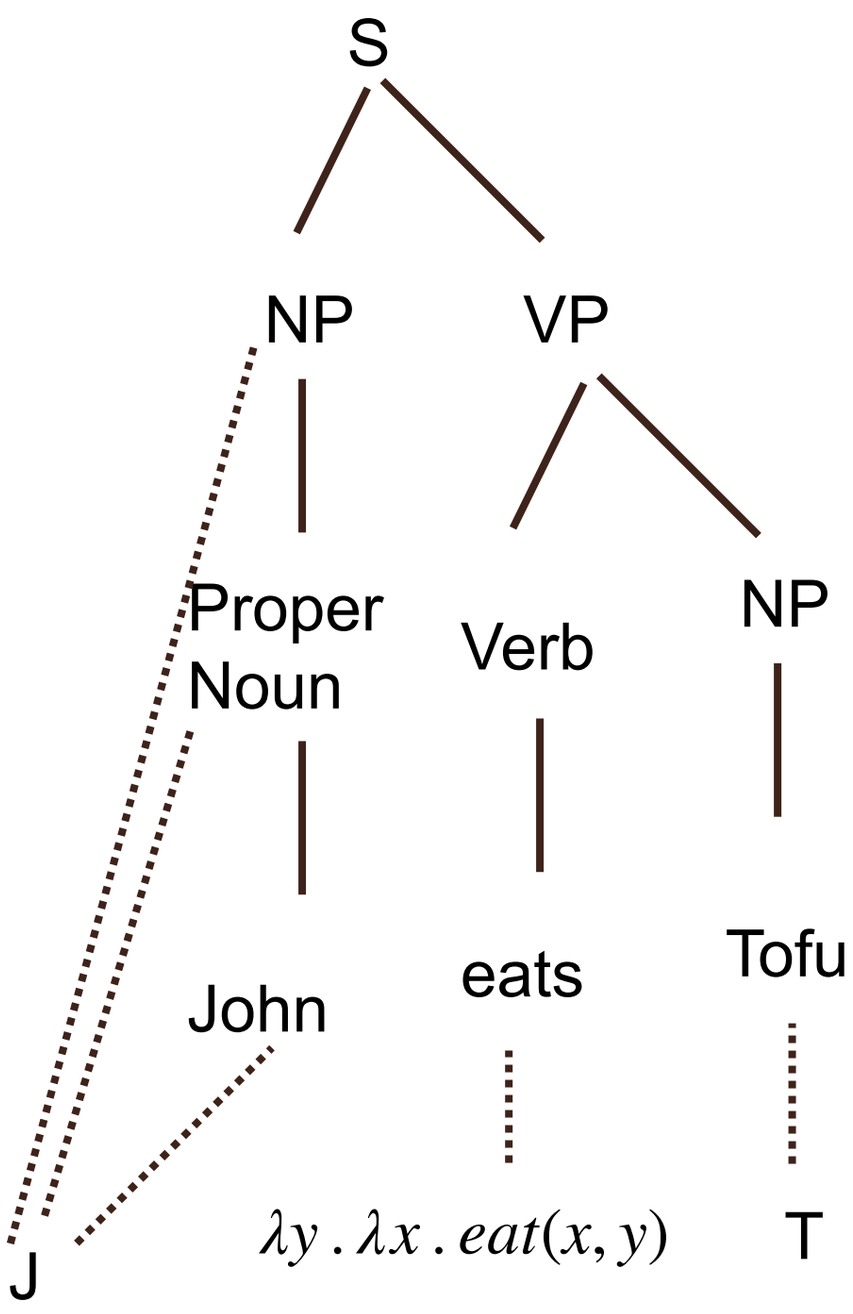
Examples



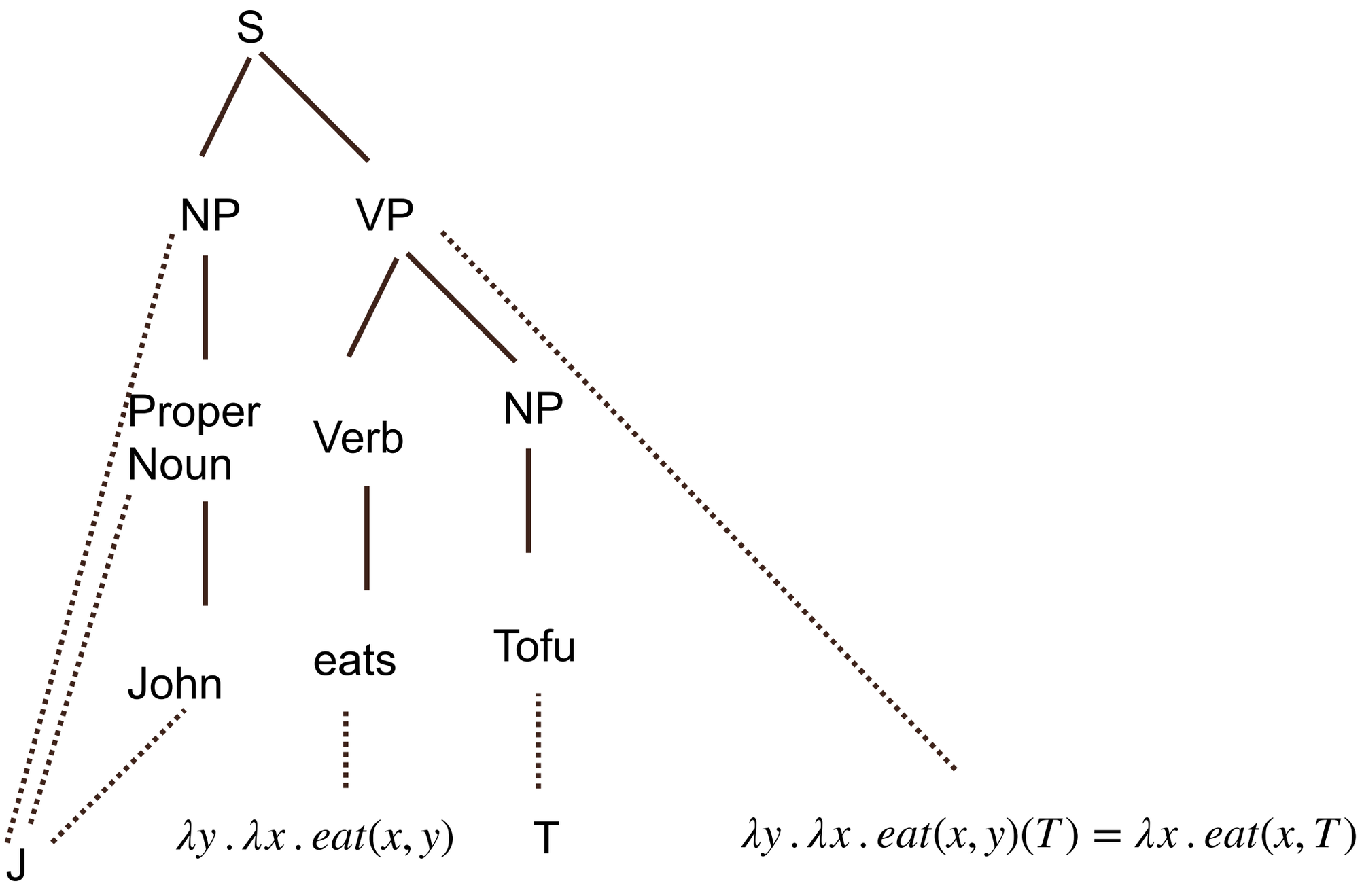
Examples


S
NP
VP
Proper
Noun
Verb
NP
John
eats
Tofu
J
$\lambda y \,.\, \lambda x \,.\, eat(x, y)$
T

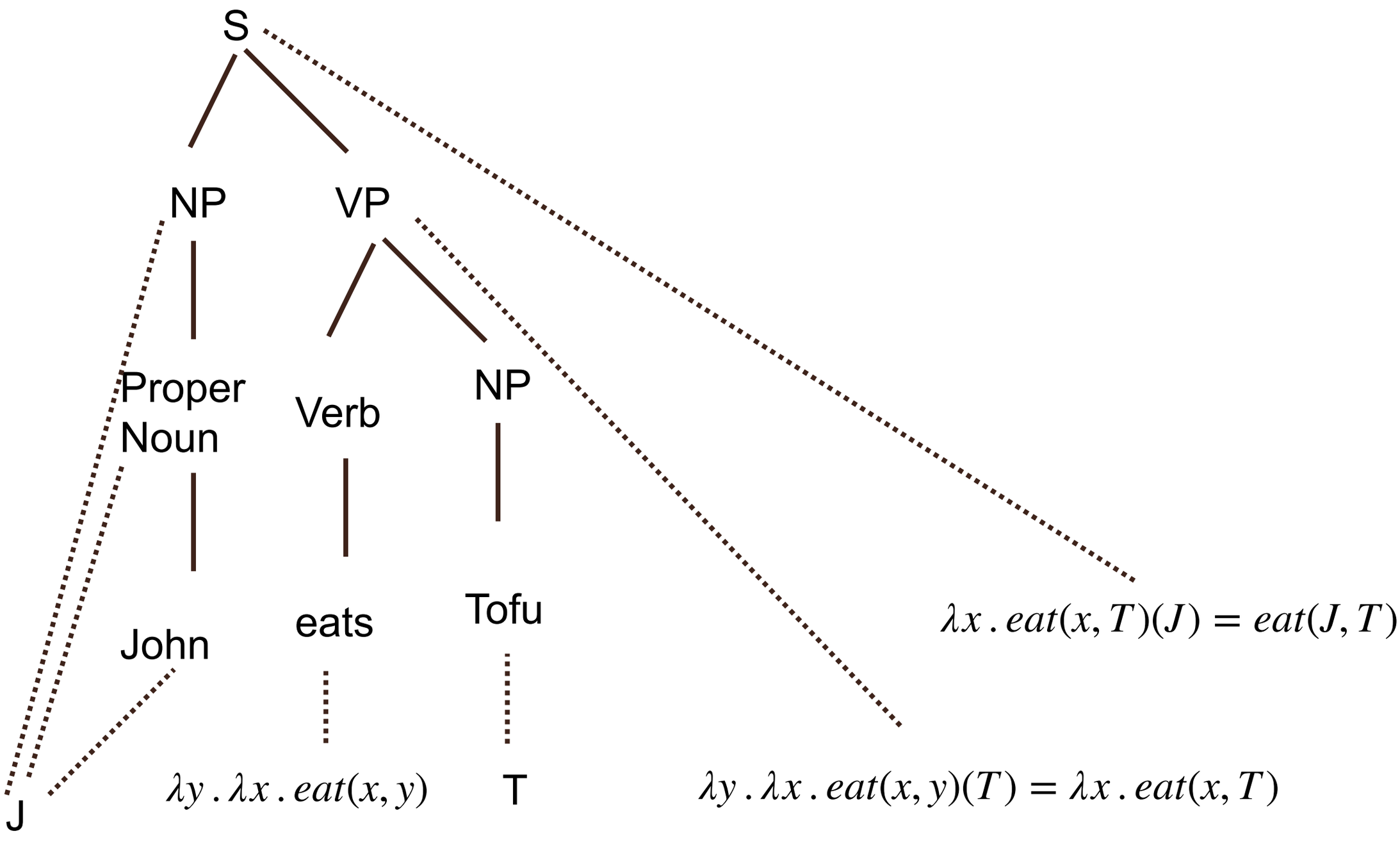
Examples


S
NP
VP
Proper
Noun
Verb
NP
John
eats
Tofu
J
$\lambda y \,.\, \lambda x \,.\, eat(x, y)$
T

Examples

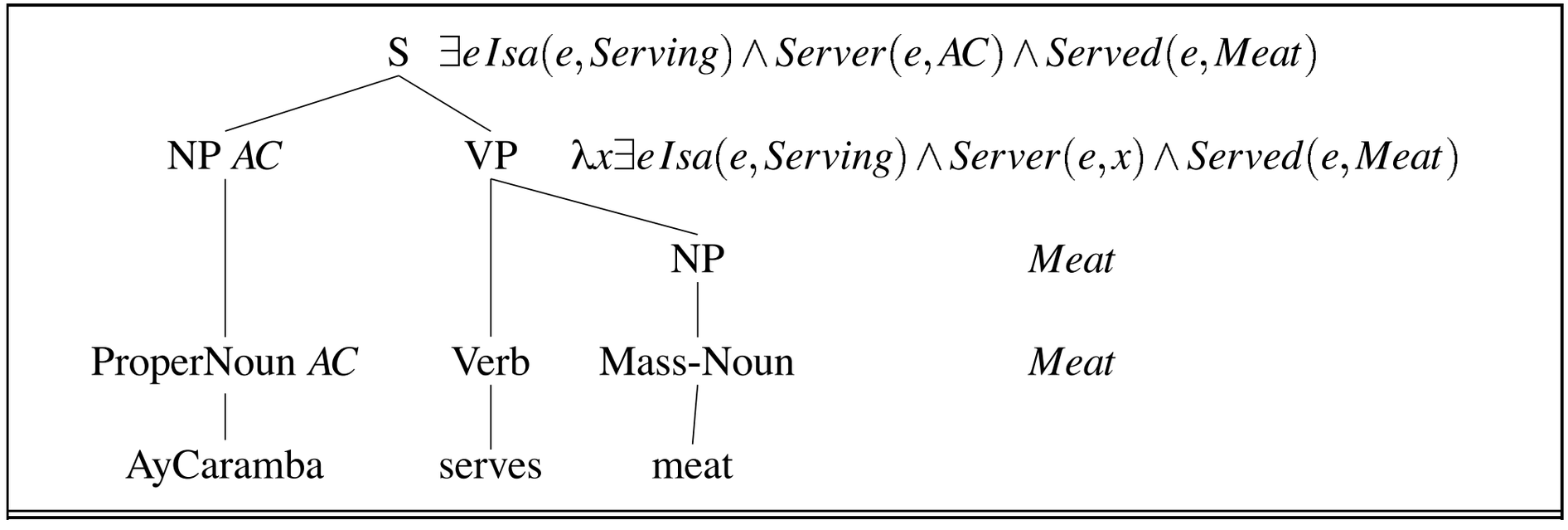

S
NP
VP
Proper Noun
Verb
NP
John
eats
Tofu
J
$\lambda y\,.\,\lambda x\,.\,eat(x,y)$
T
$\lambda y\,.\,\lambda x\,.\,eat(x,y)(T) = \lambda x\,.\,eat(x,T)$

Examples


S
NP
VP
Proper Noun
Verb
NP
John
eats
Tofu
J
$\lambda y \,.\, \lambda x \,.\, eat(x, y)$
T
$\lambda y \,.\, \lambda x \,.\, eat(x, y)(T) = \lambda x \,.\, eat(x, T)$
$\lambda x \,.\, eat(x, T)(J) = eat(J, T)$

In reality, things are more complicated.

- The semantic analyser assigns much more complicated semantics to verbs, in order to cover as much as possible of their meaning.
- Thus the semantics of the sentence also becomes more complicated. But the computational principles are the same.



OUTLINE

1) Semantics
2) First Order Logic and Semantic Models
3) Higher-order logic and mapping syntax to semantics
4) Distributional Hypothesis for meaning
5) Vector-space semantics
6) Smoothing parameters and similarity measures

Distributional Hypothesis

- Words that occur in similar contexts tend to have similar meanings. This insight was first formulated by Harris (1954) who said:
  - "oculist and eye-doctor . . . occur in almost the same environments"
- and more generally that
  - "If A and B have almost identical environments. . . we say that they are synonyms."

- The most famous statement of the principle comes a few years later from the linguist Firth (1957), who phrased it as

  - "You shall know a word by the company it keeps!"

- The meaning of a word is thus related to the **distribution** of the words around it.

Let’s test it!

- Imagine you had never seen the word *tesguino*, *but given the following text:*

A bottle of tesguino is on the table.
Everybody likes tesguino .
Tesguino makes you drunk.
We make tesguino out of corn.

Can you *guess what it means?:*

a fermented drink similar to beer made of corn

Automation

- This intuition can be automatised by counting words in the context of tesguino. These will tend to be words like:
    - bottle and drunk, which are the same as the words around:
    - beer or liquor or tequila.

- Thus, same contexts around words, help us discover the degree of similarity between words.
    - e.g. between beer and liquor and tesguino.

- A more advanced method takes into account more sophisticated features of the context, e.g. syntactic features , e.g. ‘occurs *after* drunk’, ’occurs *before* bottle’, *‘is the direct object of* ‘drink’. etc.

Distributional/Vector Semantics

- The meaning of a word is computed from the distribution of words around it, represented as a vector of numbers, related to frequency counts of the word in context.

- Main application is to **compute semantic similarity** between words, sentences, documents.

- A vast domain of NLP tasks:
  - named entity extraction, parsing, semantic role labelling, relation extraction.

- An important tool in applications like paraphrasing, question answering, summarization, automatic essay grading, etc.

OUTLINE

1) Semantics
2) First Order Logic and Semantic Models
3) Higher-order logic and mapping syntax to semantics
4) Distributional Hypothesis for meaning
5) Vector-space semantics
6) Smoothing parameters and similarity measures

Basic Linear Algebra

- A vector, e.g. v, w is a list/array of numbers, e.g.

  [1, 3, 37, 5] [1, 2, 58, 117] [8,12, 1, 0] [15,0,0,0]

- A vector space, e.g. V, W is a collection of vectors satisfying certain properties, e.g. being closed under addition and scalar multiplication.

- The entries of vectors are not arbitrary. They correspond to a quantity with respect to a basis, e.g. 1, 1, 8, 15 are the quantities of the above vectors with respect to basis 1 of the vector space.

- The number of basis of a vector space is called its dimension, it is denoted by |V|, |W|, etc, e.g. the above vector space has dimension 4.

Term-Document Matrix

- **Vector or distributional models** of meaning are generally based on a **co-occurrence matrix,** a way of representing how often words co-occur. We'll look at two popular matrices: the **term-document matrix** and the **term-term (or word-context)** matrix.

- The first vector space model developed for **information retrieval** by Salton in 1971, inventing the term-document matrix.

- **Documents** are represented by **vectors** in a vector space whose basis are words.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	5	117	0	0

Figure 15.1 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

Term-Document Matrix

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	5	117	0	0

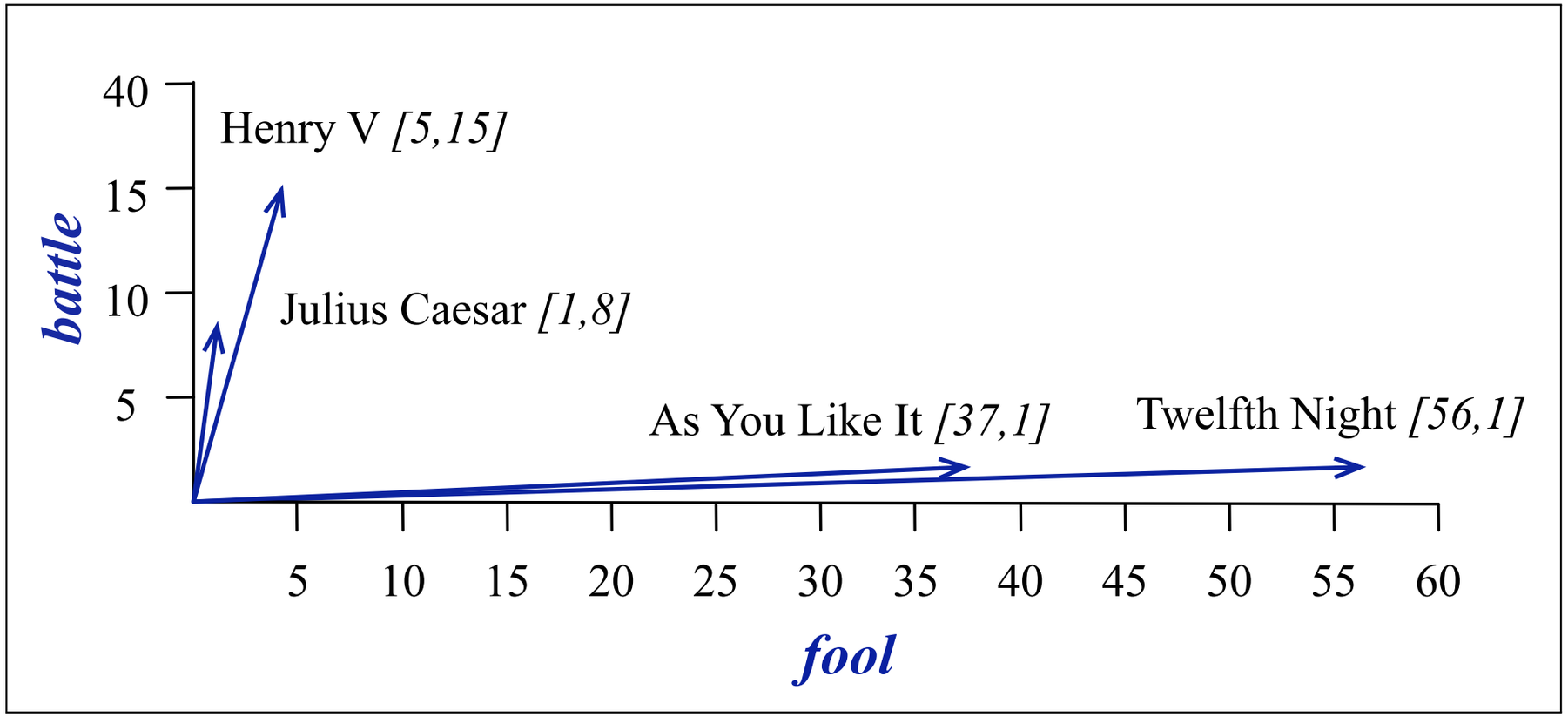
Figure 15.1 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

- Each play is represented by a vector:

  As You like It: [1, 2, 37, 5]
  Twelfth Night: [1, 2, 58, 117]
  Julius Caesar: [8, 12, 1, 0]
  Henry V: [15, 36, 5, 0]

Term-Document Matrix

- A vector v in the vector space V can be thought of as a point in the |V| dimensional space. Our Shakespeare plays are points in the 4 dimensional space.
- A **visualisation/depiction** of these vectors, with only two of the dimensions (battle and fool) is as follows:



What is it good for?

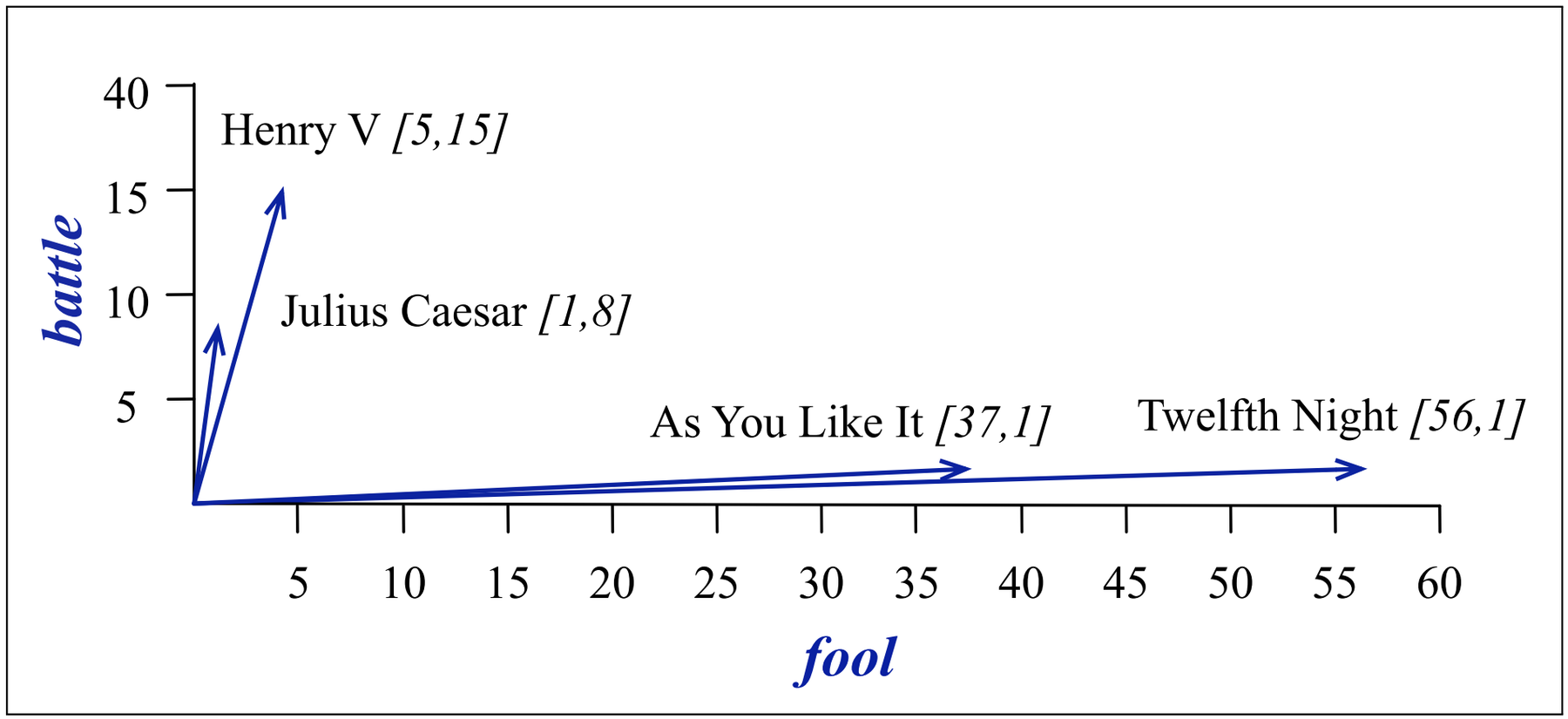
- For the task of **finding similar documents:**
  Two documents are similar iff they contain similar words.

- The geometric or **angular distance** between vectors is used to represent the **similarity** between them. The **geometric distance** between document vectors is used to compute the similarities between documents through the **cosine of the angle.**

$$\text{cosine}(\mathbf{v},\mathbf{w}) = \frac{\mathbf{v}\cdot\mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

- In **information retrieval (IR),** the distance between the vector of a query and the vector of the documents is computers and similar documents to query are retrieved as the result of search.

What is it good for?



- *As You Like It* and *Twelfth Night* have a small angle between them and so do *Julius Caesar* and *Henry V* (cosine of angle close to 1).

- *As You Like It* and *Twelfth Night* are both further away from *Julius Cesar* and *Henry V (*cosine of angle much smaller, closer to 0).

Word-Context Matrix

- A different approach to distributional meaning, focussing on **word meaning** where words which occur in **similar contexts** are deemed similar.

- Columns and rows are both labelled by words but **words** along the side are the **target words** we are interested in analysing, and the words along the **top** are **context words** which give the target words their **distributional meaning**.

- When using raw counts, each cell represents the number of times a target word (label of the row) occurs **in the context/neighbourhood of** a context word (label of column).

	aardvark	...	computer	data	pinch	result	sugar	...
apricot	0	...	0	0	1	0	1	
pineapple	0	...	0	0	1	0	1	
digital	0	...	2	1	0	1	0	
information	0	...	1	6	0	4	0	

Word-Context Matrix

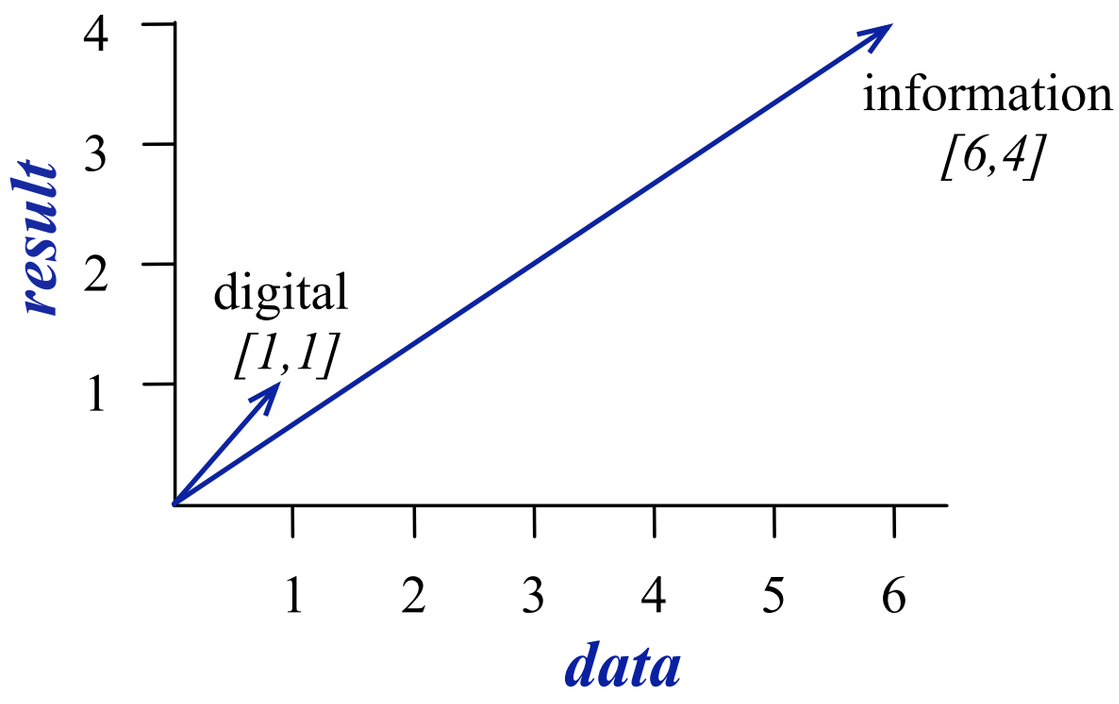
- Being in a context means being in a window of k-words around it.
- For example, here are 7-word windows around four words of the Brown corpus:

sugar, a sliced lemon, a tablespoonful of	**apricot**	preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	**pineapple**	and another fruit whose taste she likened
well suited to programming on the digital	**computer**.	In finding the optimal R-stage policy from
for the purpose of gathering data and	**information**	necessary for the study authorized in the

Word-Context Matrix

sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened
well suited to programming on the digital **computer**. In finding the optimal R-stage policy from
for the purpose of gathering data and **information** necessary for the study authorized in the

	aardvark	...	**computer**	**data**	**pinch**	**result**	**sugar**	...
apricot	0	...	0	0	1	0	1	
pineapple	0	...	0	0	1	0	1	
digital	0	...	2	1	0	1	0	
information	0	...	1	6	0	4	0	



OUTLINE

1) Semantics
2) First Order Logic and Semantic Models
3) Higher-order logic and mapping syntax to semantics
4) Distributional Hypothesis for meaning
5) Vector-space semantics
6) Smoothing parameters and similarity measures

Some Parameters of the Model

- The dimension of the vector space is the size of the vocabulary of the corpus, e.g. 10,000 or 50,000.

- This means that, lots of cells of the matrix will be 0 leading to sparse vectors, so one can use efficient sparse matrix computation algorithms to compute with these vectors.

- Size of the context window depends on the task, it is some where between 1 and 8.

- The smaller the window, the more syntactic information the co-occurrence counts represent, the longer the window, the more semantic information.

Some Parameters of the Model

- The raw frequency that the cells of a word-context matrix record, is not the best measure of association between the words.
  - We want a measure that can distinguish between very common words such as `it, the, a, they, …'.
  - We want a measure by which informative context words are weighed higher than non-informative ones.

- Examples of such measures are:
    - Pointwise Mutual Information (Fano 1961)
    - Mutual Information (Church and Hanks 1989)

$$\log_2 \frac{P(x,y)}{P(x)P(y)}$$

$$\sum_x \sum_y P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)}$$

Pointwise Mutual Information

- Pointwise mutual information between
- two events x,y tells us how often they
- occurred together in comparison to
- our expectation if the they were independent.

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- For word-context matrices, PMI is defined with regards to the occurrence between a word and a context word.

$$\text{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$$

How often w and c were observed together

How often we expect w and c to occur independently

How much they occurred together more than we expect by chance.

Positive Pointwise Mutual Information

- A logarithm can return negative values.
- In this case, this happens when PMI(w,c) <0.
- That is, when w occurred with c, less frequent than chance.
- Which is unrealistic and returns unreliable quantities for word vectors.

- So people consider Positive PMI or PPMI.

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

Computing PPMI

Given a word-context matrix F with W rows and C columns. Take f_{ij} to be the number of time word w_i occurred in the context of word c_j.

$$P(w_i,c_j) = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$P(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$P(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

Computing PPMI

Given a word-context matrix F with W rows and C columns. Take f_{ij} to be the number of time word w_i occurred in the context of word c_j.

$$P(\text{w=information},\text{c=data}) = \frac{4}{9}$$

$$P(w_i,c_j) = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$P(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$P(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

Computing PPMI

Given a word-context matrix F with W rows and C columns.
Take f_{ij} to be the number of time word w_i occurred in the context of word c_j.

$$P(\text{w=information},\text{c=data}) = \frac{4}{19} = \frac{9}{19} = .316$$

$$P(w_i,c_j) = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

f49 : number of times word 4 (information) has context word 9 (data) in its context

$$P(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$P(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

Computing PPMI

Given a word-context matrix F with W rows and C columns.
Take f_{ij} to be the number of time word w_i occurred in the context of word c_j.

$$P(\text{w=information},\text{c=data}) = \frac{4}{19} = \frac{9}{19} = .316$$

$$P(w_i,c_j) = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

f11+ f12+ … + f21+ f22+ … f31+f32+ …= no of words in the corpus.

$$P(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$P(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

Computing PPMI

Given a word-context matrix F with W rows and C columns.
Take f_{ij} to be the number of time word w_i occurred in the context of word c_j.

$$P(w_i,c_j) = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

f41+ f42+f43+…= no of times word 4 occurred in any context.

$$P(\text{w=information}) = \frac{11}{19} = .579$$

$$P(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$P(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

Computing PPMI

Given a word-context matrix F with W rows and C columns.
Take f_{ij} to be the number of time word w_i occurred in the context of word c_j.

$$P(\text{w=information,c=data}) = \frac{6}{19} = .316$$

$$P(w_i,c_j) = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$P(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

f19+f29+f39+…= no. of times any word occurred with context 9.

$$P(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$P(\text{c=data}) = \frac{7}{19} = .368$$

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

Computing PPMI

Given a word-context matrix F with W rows and C columns.
Take f_{ij} to be the number of time word w_i occurred in the context of word c_j.

$$P(\text{w=information,c=data}) = \frac{6}{19} = .316$$

$$P(w_i,c_j) = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$P(\text{w=information}) = \frac{11}{19} = .579$$

$$P(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$P(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$P(\text{c=data}) = \frac{7}{19} = .368$$

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

$$\log 2(.316/(.368 * .579)) = .568$$

Comparison

Raw Frequencies

	aardvark	...	computer	data	pinch	result	sugar	...
apricot	0	...	0	0	1	0	1	
pineapple	0	...	0	0	1	0	1	
digital	0	...	2	1	0	1	0	
information	0	...	1	6	0	4	0	

PPMI

	computer	data	pinch	result	sugar
apricot	0	0	2.25	0	2.25
pineapple	0	0	2.25	0	2.25
digital	1.66	0	0	0	0
information	0	0.57	0	0.47	0

$$\log 2(.05/(.16 * .58)) = -0.618$$

Smoothing

- PMI has the problem of being biased toward infrequent events; very rare words tend to have very high PMI values. A solution is obtained by smoothing.
- Laplace Smoothing: before computing PMI, add a small constant k (e.g. 1, 2, 3) to all counts. This increases the 0 counts and shrinks the non zero counts.

	computer	data	pinch	result	sugar
apricot	2	2	3	2	3
pineapple	2	2	3	2	3
digital	4	3	2	3	2
information	3	8	2	6	2

Figure 15.8 Laplace (add-2) smoothing of the counts in Fig. 15.4.

	computer	data	pinch	result	sugar
apricot	0	0	0.56	0	0.56
pineapple	0	0	0.56	0	0.56
digital	0.62	0	0	0	0
information	0	0.58	0	0.37	0

Figure 15.9 The Add-2 Laplace smoothed PPMI matrix from the add-2 smoothing counts in Fig. 15.8.

Smoothing

- Another method (Levy et al 2015) is to raise the probability of context to a power.

$$\text{PPMI}_\alpha(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0)$$

$$P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha}$$

  - Experiments have shown that raising to the power 0.75 betters the results.

- This works because raising the probability to α = 0.75 increases the probability assigned to rare contexts, and hence lowers their PMI (Pα(c) > P(c) when c is rare).

Other measures

- TF-IDF: from Information Retrieval.
  - TF: frequency of the word in the document (Luhn 1975),
  - IDF: inverse document frequency (Sparck Jones, 1972).

- It is one way of assigning higher weights to the more discriminative words.

- IDF is defined using the fraction **N/df_i** where N is the total number of documents in the collection, and dfi is the number of documents in which term i occurs.

- The fewer documents in which a term occurs, the higher this weight.

- Because of the large number of documents in many collections, this measure is usually squashed with a log function.

$$\text{idf}_i = \log\left(\frac{N}{df_i}\right)$$

Measuring Similarity

- The geometric distance between vectors of words.
- If vectors have length 1, this is their dot product:

$$\text{dot-product}(\vec{v},\vec{w}) = \vec{v}\cdot\vec{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

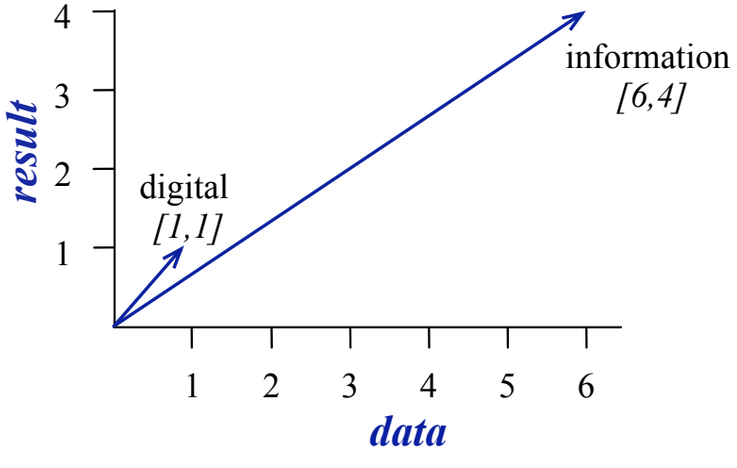
- If not, it is the cosine between the two vectors.

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

<u>Other measures (read in Ch. 6 of J&M):</u>

- Jaccard and Dice, from Information Retrieval.
- Kullback-Leibler Divergence and Shannon-Jenson from relative entropy in Information Theory.

Cosine distance for similarity



What is the cosine distance between these vectors?:

$$\cos\alpha = \frac{\bar{a}\cdot\bar{b}}{|\bar{a}||\bar{b}|}$$

1. For numerator, get the dot product of the two vectors: $\bar{a}\cdot\bar{b} = a_x\cdot b_x + a_y\cdot b_y =$ 1 * 6 + 1 * 4 = 10

2. For denominator, get the product of the magnitudes:

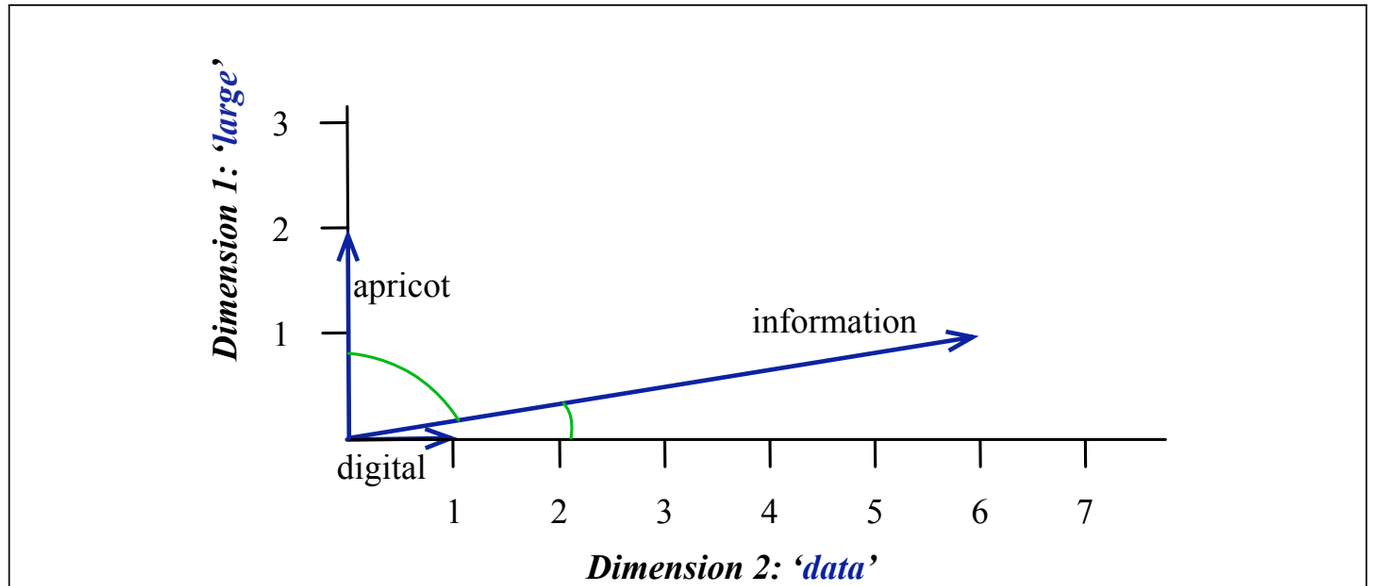
$$|\bar{a}| = \sqrt{a_x^2 + a_y^2} = \sqrt{1^2+1^2} = 1.414$$
$$|\bar{b}| = \sqrt{b_x^2 + b_y^2} = \sqrt{6^2+4^2} = 7.211$$

3. Put it together:

$$\cos\alpha = \frac{\bar{a}\cdot\bar{b}}{|\bar{a}||\bar{b}|} = \frac{10}{1.414 * 7.211} = \frac{10}{10.198} = 0.981$$

Examples of Similarity

	large	data	computer
apricot	2	0	0
digital	0	1	2
information	1	6	1



$$\cos(\text{apricot}, \text{information}) = \frac{2+0+0}{\sqrt{4+0+0}\sqrt{1+36+1}} = \frac{2}{2\sqrt{38}} = .16$$

$$\cos(\text{digital}, \text{information}) = \frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = \frac{8}{\sqrt{38}\sqrt{5}} = .58$$

Syntactic Vector Models

- Instead of defining a word's context by nearby words, we can define it by syntactic relations.
- The meaning of entities and the meaning of grammatical relations among them is related to their restrictions to other entities. (Harris 1968)
- The similarity between meanings of duty and responsibility is mirrored in their syntactic behaviour.
- They both can be modified by adjectives: additional, administrative, collective, congressional;
- They both can be the direct objects of verbs: assert, assign, assume, attend to avoid, become, etc.

	subj-of, absorb	*subj-of*, adapt	*subj-of*, behave	...	*pobj-of*, inside	*pobj-of*, into	...	*nmod-of*, abnormality	*nmod-of*, anemia	*nmod-of*, architecture	...	*obj-of*, attack	*obj-of*, call	*obj-of*, come from	*obj-of*, decorate	...	*nmod*, bacteria	*nmod*, body	*nmod*, bone marrow
cell	1	1	1		16	30		3	8	1		6	11	3	2		3	2	2

Evaluating Vector Models

- WordSim-353 : a set of of ratings for 353 noun pairs.
- SimLex-999 : quantifies similarity (cup, mug) rather than relatedness (cup, coffee), and includes adjective, noun and verb pairs.
- TOEFL dataset : a set of 80 questions, each consisting of a target word with 4 additional word choices; the task is to
- choose which is the correct synonym.
- Stanford Contextual Word Similarity (SCWS) : human similarity ratings for 2,003 pairs of words, but in their sentential context.
- Word2Vec Dataset: a set of patterns “a is to b as c is to d”. Given a, b, and c, the task is to find d. The words are in certain semantic relationships with each other. For example Athens is to Greece as Oslo is to ? …. Norway.: Mikolov et al. 2013.

Reading

Jurafsky and Martin (3rd Ed):

- Chapter 15 (Logical semantics)
- Chapter 6 (Distributional/Vector Space Semantics)