Logics and Statistics for Language Modeling

Carlos Areces

areces@loria.fr
http://www.loria.fr/~areces/ls
INRIA Nancy Grand Est
Nancy, France

2009/2010

Today's Program

Today's Program

- CNF in a clever way.
- Incomplete Methods for PL.
  - The Greedy Algorithm and GSAT
- Inference and NLP
  - Satisfiability, Inference, Informativity
  - Some NLP phenomena that requires inference

Conjunctive Normal Form

Conjunctive Normal Form

- Write φ in conjunctive normal form (CNF)

$$\varphi = \bigwedge_{l \in L} \bigvee_{m \in M} \psi_{(l,m)}, \psi \text{ a literal (i.e., } p \text{ or } \neg p).$$

This just means: No conjunctions inside disjunctions
Negations only on propositional symbols

- Using the following equivalences:

$$\begin{array}{rcl}
(\neg(\varphi \vee \psi)) & \rightsquigarrow & (\neg\varphi \wedge \neg\psi) \\
(\neg(\varphi \wedge \psi)) & \rightsquigarrow & (\neg\varphi \vee \neg\psi) \\
(\neg\neg\varphi) & \rightsquigarrow & \varphi \\
(\varphi \vee (\psi \wedge \theta)) & \rightsquigarrow & ((\varphi \vee \psi) \wedge (\varphi \vee \theta)) \\
((\psi \wedge \theta) \vee \varphi) & \rightsquigarrow & ((\varphi \vee \psi) \wedge (\varphi \vee \theta))
\end{array}$$

Conjunctive Normal Form

Conjunctive Normal Form

- This conversion to CNF can lead to exponentially big formulas. Consider

$$(p_1 \wedge q_1) \vee (p_2 \wedge q_2) \vee \cdots \vee (p_n \wedge q_n).$$

Conjunctive Normal Form

- This conversion to CNF can lead to exponentially big formulas. Consider

$$(p_1 \wedge q_1) \vee (p_2 \wedge q_2) \vee \cdots \vee (p_n \wedge q_n).$$

- In CNF we get a formula:

$$(p_1 \vee \cdots \vee p_{n-1} \vee p_n) \wedge (p_1 \vee \cdots \vee p_{n-1} \vee q_n) \wedge \cdots \wedge (q_1 \vee \cdots \vee q_{n-1} \vee q_n).$$

Conjunctive Normal Form

- This conversion to CNF can lead to exponentially big formulas. Consider

$$(p_1 \wedge q_1) \vee (p_2 \wedge q_2) \vee \cdots \vee (p_n \wedge q_n).$$

- In CNF we get a formula:

$$(p_1 \vee \cdots \vee p_{n-1} \vee p_n) \wedge (p_1 \vee \cdots \vee p_{n-1} \vee q_n) \wedge \cdots \wedge (q_1 \vee \cdots \vee q_{n-1} \vee q_n).$$

- Which has 2^n clauses: each clause contains either p_i or q_i.

Conjunctive Normal Form

- This conversion to CNF can lead to exponentially big formulas. Consider

$$(p_1 \wedge q_1) \vee (p_2 \wedge q_2) \vee \cdots \vee (p_n \wedge q_n).$$

- In CNF we get a formula:

$$(p_1 \vee \cdots \vee p_{n-1} \vee p_n) \wedge (p_1 \vee \cdots \vee p_{n-1} \vee q_n) \wedge \cdots \wedge (q_1 \vee \cdots \vee q_{n-1} \vee q_n).$$

- Which has 2^n clauses: each clause contains either p_i or q_i.
- We can obtain formulas en CNF which are only polynomially bigger than the original formula. But they are only equisatisfiable to the input and not equivalent.

CNF: Using New Propositional Symbols

CNF: Using New Propositional Symbols

- Consider again

$$\varphi = (p_1 \wedge q_1) \vee (p_2 \wedge q_2) \vee \cdots \vee (p_n \wedge q_n).$$

CNF: Using New Propositional Symbols

- Consider again

$$\varphi = (p_1 \wedge q_1) \vee (p_2 \wedge q_2) \vee \cdots \vee (p_n \wedge q_n).$$

- We can write φ' as:

$$(r_1 \vee \cdots \vee r_n) \wedge (\neg r_1 \vee p_1) \wedge (\neg r_1 \vee q_1) \wedge \cdots \wedge (\neg r_n \vee p_n) \wedge (\neg r_n \vee q_n).$$

CNF: Using New Propositional Symbols

- Consider again

$$\varphi = (p_1 \wedge q_1) \vee (p_2 \wedge q_2) \vee \cdots \vee (p_n \wedge q_n).$$

- We can write φ' as:

$$(r_1 \vee \cdots \vee r_n) \wedge (\neg r_1 \vee p_1) \wedge (\neg r_1 \vee q_1) \wedge \cdots \wedge (\neg r_n \vee p_n) \wedge (\neg r_n \vee q_n).$$

- A model satisfies φ' if at least one of the new variables r_i is true. If r_i is true, then p_i and q_i are true:
  Every model that satisfies the translation also satisfies φ.

CNF: Using New Propositional Symbols

- Consider again

$$\varphi = (p_1 \wedge q_1) \vee (p_2 \wedge q_2) \vee \cdots \vee (p_n \wedge q_n).$$

- We can write φ' as:

$$(r_1 \vee \cdots \vee r_n) \wedge (\neg r_1 \vee p_1) \wedge (\neg r_1 \vee q_1) \wedge \cdots \wedge (\neg r_n \vee p_n) \wedge (\neg r_n \vee q_n).$$

- A model satisfies φ' if at least one of the new variables r_i is true. If r_i is true, then p_i and q_i are true:
  Every model that satisfies the translation also satisfies φ.
- On the other hand, if we have a model for φ then it makes some p_i and q_i true. We can get a model for φ' by seting r_i true.

Approximation Methods: Motivation

Approximation Methods: Motivation

- DP can reliably solve problems with up to 500 variables. . .

Approximation Methods: Motivation

- DP can reliably solve problems with up to 500 variables. . .
- . . . but problems commonly arising in practice often need 1000s of variables

Approximation Methods: Motivation

- DP can reliably solve problems with up to 500 variables...
- ... but problems commonly arising in practice often need 1000s of variables
- Depending on the application, semi-decision procedures can be useful: find a solution if one exists

Approximation Methods: Motivation

- DP can reliably solve problems with up to 500 variables...
- ... but problems commonly arising in practice often need 1000s of variables
- Depending on the application, semi-decision procedures can be useful: find a solution if one exists
- E.g., plan existence $\equiv$ model finding, and we may not be interested in non-existence of a plan

Approximation Methods: Motivation

- DP can reliably solve problems with up to 500 variables...
- ... but problems commonly arising in practice often need 1000s of variables
- Depending on the application, semi-decision procedures can be useful: find a solution if one exists
- E.g., plan existence $\equiv$ model finding, and we may not be interested in non-existence of a plan
- Finally, we might need "anytime answers" which can provide a "best guess" at any point we stop the algorithm

Flipping Coins: The “Greedy” Algorithm

Flipping Coins: The “Greedy” Algorithm

- This algorithm is due to Koutsopias and Papadimitriou
  - main idea: flip variables till you can no longer increase the number of satisfied clauses

```
procedure greedy(Sigma)
```

Flipping Coins: The "Greedy" Algorithm

- This algorithm is due to Koutsopias and Papadimitriou
  - main idea: flip variables till you can no longer increase the number of satisfied clauses

```
procedure greedy(Sigma)
  T := random(Sigma) // random assignment
```

Flipping Coins: The "Greedy" Algorithm

- This algorithm is due to Koutsopias and Papadimitriou
  - main idea: flip variables till you can no longer increase the number of satisfied clauses

```
procedure greedy(Sigma)
  T := random(Sigma) // random assignment
  repeat until no improvement possible
```

Flipping Coins: The "Greedy" Algorithm

- This algorithm is due to Koutsopias and Papadimitriou
  - main idea: flip variables till you can no longer increase the number of satisfied clauses

```
procedure greedy(Sigma)
  T := random(Sigma) // random assignment
  repeat until no improvement possible
    T := T with variable flipped that increases
         the number of satisfied clauses
```

Flipping Coins: The "Greedy" Algorithm

- This algorithm is due to Koutsopias and Papadimitriou
  - main idea: flip variables till you can no longer increase the number of satisfied clauses

```
procedure greedy(Sigma)
  T := random(Sigma) // random assignment
  repeat until no improvement possible
    T := T with variable flipped that increases
         the number of satisfied clauses
  end
```

The GSAT Procedure

The GSAT Procedure

- This algorithm is due to Selman, Levesque and Mitchell
  - Adds restarts to the simple "greedy" algorithm, and also allows sideways flips (not necessarily increasing the "cost function.")

The GSAT Procedure

- This algorithm is due to Selman, Levesque and Mitchell
  - Adds restarts to the simple "greedy" algorithm, and also allows sideways flips (not necessarily increasing the "cost function.")

```
procedure GSAT(Sigma)
```

The GSAT Procedure

- This algorithm is due to Selman, Levesque and Mitchell
  - Adds restarts to the simple “greedy” algorithm, and also allows sideways flips (not necessarily increasing the “cost function.”)

```
procedure GSAT(Sigma)
  for i := 1 to MAX-TRIES // These are the restarts
```

The GSAT Procedure

- This algorithm is due to Selman, Levesque and Mitchell
  - Adds restarts to the simple "greedy" algorithm, and also allows sideways flips (not necessarily increasing the "cost function.")

```
procedure GSAT(Sigma)
  for i := 1 to MAX-TRIES // These are the restarts
    T := random(Sigma) // random assignment
```

The GSAT Procedure

- This algorithm is due to Selman, Levesque and Mitchell
  - Adds restarts to the simple "greedy" algorithm, and also allows sideways flips (not necessarily increasing the "cost function.")

```
procedure GSAT(Sigma)
  for i := 1 to MAX-TRIES // These are the restarts
    T := random(Sigma) // random assignment
    for j := 1 to MAX-FLIPS // To ensure termination
```

The GSAT Procedure

- This algorithm is due to Selman, Levesque and Mitchell
  - Adds restarts to the simple "greedy" algorithm, and also allows sideways flips (not necessarily increasing the "cost function.")

```
procedure GSAT(Sigma)
  for i := 1 to MAX-TRIES // These are the restarts
    T := random(Sigma) // random assignment
    for j := 1 to MAX-FLIPS // To ensure termination
      if T satisfies Sigma then return T
```

The GSAT Procedure

- This algorithm is due to Selman, Levesque and Mitchell
  - Adds restarts to the simple "greedy" algorithm, and also allows sideways flips (not necessarily increasing the "cost function.")

```
procedure GSAT(Sigma)
  for i := 1 to MAX-TRIES // These are the restarts
    T := random(Sigma) // random assignment
    for j := 1 to MAX-FLIPS // To ensure termination
      if T satisfies Sigma then return T
      else T := T with variable flipped to maximize
                  number of satisfied clauses
```

The GSAT Procedure

- This algorithm is due to Selman, Levesque and Mitchell
  - Adds restarts to the simple "greedy" algorithm, and also allows sideways flips (not necessarily increasing the "cost function.")

```
procedure GSAT(Sigma)
  for i := 1 to MAX-TRIES // These are the restarts
    T := random(Sigma) // random assignment
    for j := 1 to MAX-FLIPS // To ensure termination
      if T satisfies Sigma then return T
      else T := T with variable flipped to maximize
                number of satisfied clauses
      // It doesn't matter if the number does
      // not increase. These are the sideways flips
```

The GSAT Procedure

- This algorithm is due to Selman, Levesque and Mitchell
  - Adds restarts to the simple "greedy" algorithm, and also allows sideways flips (not necessarily increasing the "cost function.")

```
procedure GSAT(Sigma)
  for i := 1 to MAX-TRIES // These are the restarts
    T := random(Sigma) // random assignment
    for j := 1 to MAX-FLIPS // To ensure termination
      if T satisfies Sigma then return T
      else T := T with variable flipped to maximize
                  number of satisfied clauses
      // It doesn't matter if the number does
      // not increase. These are the sideways flips
    end
  end
```

GSAT: Evaluation

GSAT: Evaluation

- The GSAT procedure has been highly influential

GSAT: Evaluation

- The GSAT procedure has been highly influential
- GSAT is very good on some kinds of problems (random 3-SAT, *n*-queens, etc.)

GSAT: Evaluation

- The GSAT procedure has been highly influential
- GSAT is very good on some kinds of problems (random 3-SAT, *n*-queens, etc.)

formulas	
var	clauses
50	215
100	430
140	602
150	645
300	1275
500	2150

GSAT: Evaluation

- The GSAT procedure has been highly influential
- GSAT is very good on some kinds of problems (random 3-SAT, *n*-queens, etc.)

formulas		GSAT		
var	clauses	M-FLIPS	restarts	time
50	215	250	6.4	0.4s
100	430	500	42.5	6s
140	602	700	52.6	14s
150	645	1500	100.5	45s
300	1275	6000	231.8	12m
500	2150	10000	995.8	1.6h

GSAT: Evaluation

- The GSAT procedure has been highly influential
- GSAT is very good on some kinds of problems (random 3-SAT, *n*-queens, etc.)

formulas		GSAT			DP		
var	clauses	M-FLIPS	restarts	time	choices	depth	time
50	215	250	6.4	0.4s	77	11	1.4s
100	430	500	42.5	6s	84×10^3	19	2.8m
140	602	700	52.6	14s	2.2×10^6	27	4.7h
150	645	1500	100.5	45s	–	–	–
300	1275	6000	231.8	12m	–	–	–
500	2150	10000	995.8	1.6h	–	–	–

GSAT: Sideway Moves

GSAT: Sideway Moves

- Recall: one of the differences between the "greedy" algorithm and GSAT is the possibility of sideways moves
- Does this difference matter?

GSAT: Sideway Moves

- Recall: one of the differences between the "greedy" algorithm and GSAT is the possibility of sideways moves
- Does this difference matter?

type
random
random
30-queens

GSAT: Sideway Moves

- Recall: one of the differences between the "greedy" algorithm and GSAT is the possibility of sideways moves
- Does this difference matter?

type	formulas	
	vars	clauses
random	50	215
random	100	430
30-queens	900	43240

GSAT: Sideway Moves

- Recall: one of the differences between the "greedy" algorithm and GSAT is the possibility of sideways moves
- Does this difference matter?

type	formulas		M-FLIPS
	vars	clauses	
random	50	215	1000
random	100	430	100000
30-queens	900	43240	100000

GSAT: Sideway Moves

- Recall: one of the differences between the "greedy" algorithm and GSAT is the possibility of sideways moves
- Does this difference matter?

type	formulas		M-FLIPS	no sideways moves		
	vars	clauses		%-solved	restarts	time
random	50	215	1000	69 %	537	10s
random	100	430	100000	39 %	63382	15m
30-queens	900	43240	100000	100 %	50000	30h

GSAT: Sideway Moves

- Recall: one of the differences between the “greedy” algorithm and GSAT is the possibility of sideways moves
- Does this difference matter?

type	formulas		M-FLIPS	no sideways moves			all moves		
	vars	clauses		%-solved	restarts	time	%-solved	tries	time
random	50	215	1000	69 %	537	10s	100 %	6	1.4s
random	100	430	100000	39 %	63382	15m	100 %	81	2.8m
30-queens	900	43240	100000	100 %	50000	30h	100 %	1	2.5s

Walksat

- Walksat is a local search algorithm to solve satisfiability for PL, and improved version of GSAT.
- Site: `http://www.cs.rochester.edu/u/kautz/walksat`
- Walksat has been proven particularly useful in solving satisfiability problems produced by conversion from automated planning problems.

Propositional Logics: Conclusions

Propositional Logics: Conclusions

- Even in a very simple language like PL, we can encode interesting problems (e.g., graph coloring, planning, etc.)

Propositional Logics: Conclusions

- Even in a very simple language like PL, we can encode interesting problems (e.g., graph coloring, planning, etc.)
- Complete methods are guaranteed to find each and every possible solution

Propositional Logics: Conclusions

- Even in a very simple language like PL, we can encode interesting problems (e.g., graph coloring, planning, etc.)
- Complete methods are guaranteed to find each and every possible solution
- Approximation methods guarantee soundness but not completeness (i.e., if they find a solution, then indeed it is correct, but they can finish saying 'I don't know'.)

Propositional Logics: Conclusions

- Even in a very simple language like PL, we can encode interesting problems (e.g., graph coloring, planning, etc.)
- Complete methods are guaranteed to find each and every possible solution
- Approximation methods guarantee soundness but not completeness (i.e., if they find a solution, then indeed it is correct, but they can finish saying 'I don't know'.)
- Even for "simple" propositional logic things can go badly wrong if you not do things properly

Satisfiable and Unsatisfiable

Satisfiable and Unsatisfiable

- Till now, we mainly discuss the notions of satisfiability and unsatisfiability for a formula.

Satisfiable and Unsatisfiable

- Till now, we mainly discuss the notions of satisfiability and unsatisfiability for a formula.
- **Definiton:** A formula φ is satisfiable if there is a model $\mathcal{M}$ (a situation) in which it is true (notation $\mathcal{M} \models \varphi$).

Satisfiable and Unsatisfiable

- Till now, we mainly discuss the notions of satisfiability and unsatisfiability for a formula.
- **Definiton:** A formula φ is satisfiable if there is a model $\mathcal{M}$ (a situation) in which it is true (notation $\mathcal{M} \models \varphi$).
- The notion can easily be extended to sets of formulas. A set of formulas Σ is satisfiable if there is a model $\mathcal{M}$ s.t. for each $\varphi \in \Sigma$, $\mathcal{M} \models \varphi$. (Intuitively, if there is a situation in which all formulas in Σ are true.)

Satisfiable and Unsatisfiable

- Till now, we mainly discuss the notions of satisfiability and unsatisfiability for a formula.
- **Definiton:** A formula φ is satisfiable if there is a model $\mathcal{M}$ (a situation) in which it is true (notation $\mathcal{M} \models \varphi$).
- The notion can easily be extended to sets of formulas. A set of formulas Σ is satisfiable if there is a model $\mathcal{M}$ s.t. for each $\varphi \in \Sigma$, $\mathcal{M} \models \varphi$. (Intuitively, if there is a situation in which all formulas in Σ are true.)
  We can also say that Σ is consistent (we can imagine it been true in some situation).

Satisfiable and Unsatisfiable

- Till now, we mainly discuss the notions of satisfiability and unsatisfiability for a formula.
- **Definiton:** A formula φ is satisfiable if there is a model $\mathcal{M}$ (a situation) in which it is true (notation $\mathcal{M} \models \varphi$).
- The notion can easily be extended to sets of formulas. A set of formulas Σ is satisfiable if there is a model $\mathcal{M}$ s.t. for each $\varphi \in \Sigma$, $\mathcal{M} \models \varphi$. (Intuitively, if there is a situation in which all formulas in Σ are true.)
  We can also say that Σ is consistent (we can imagine it been true in some situation).
- Sometimes it's more natural to talk about other notions like inference and informativity, which can be defined in terms of satisfiability.

Inference and Informativity

Inference and Informativity

- **Definition:** Given a set of formulas Δ, we say that a formula φ is inferred from Δ (notation: $\Delta \models \varphi$), if every model that satisfies Δ, also satisfies φ.

Inference and Informativity

- **Definition:** Given a set of formulas Δ, we say that a formula φ is inferred from Δ (notation: $\Delta \models \varphi$), if every model that satisfies Δ, also satisfies φ.
- **Definition:** given a set of formulas Δ, we say that a formula φ is informative with respect to Δ, if φ is not inferred from Δ.

Inference and Informativity

- **Definition:** Given a set of formulas Δ, we say that a formula φ is inferred from Δ (notation: $\Delta \models \varphi$), if every model that satisfies Δ, also satisfies φ.
- **Definition:** given a set of formulas Δ, we say that a formula φ is informative with respect to Δ, if φ is not inferred from Δ.
- Intuitively, a sentence φ is informative with a set of other sentences Δ, if we could imagine $\Delta \cup \{\neg\varphi\}$ being satisfiable. Hence, whne we are told φ we learn something new (we eliminate some models).

Disclaimer: All mimsy were the borogoves!!!

Disclaimer: All mimsy were the borogoves!!!

Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

Disclaimer: All mimsy were the borogoves!!!

Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

Brillig: Four o'clock in the afternoon: the time when you begin broiling things for dinner.

Slithy: Combination of "slimy" and "lithe"

Tove: A combination of a badger, a lizard, and a corkscrew.

Disclaimer: All mimsy were the borogoves!!!

Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

Brillig: Four o'clock in the afternoon: the time when you begin broiling things for dinner.

Slithy: Combination of "slimy" and "lithe"

Tove: A combination of a badger, a lizard, and a corkscrew.

Inference and NLP

Inference and NLP

Let's consider the following dialogue between Ivonne and Jean:

```
I:  I've just been for tea at the Excelsior.
    The macarons des soeurs were great!
J:  Well, the benedictine nuns did know how to cook.
```

Inference and NLP

Let's consider the following dialogue between Ivonne and Jean:

```
I:  I've just been for tea at the Excelsior.
    The macarons des soeurs were great!
J:  Well, the benedictine nuns did know how to cook.
```

- Notice that some information is presupposed in this dialogue (as part of the background knowledge shared by Ivonne and Jean).

Inference and NLP

Let's consider the following dialogue between Ivonne and Jean:

```
I:  I've just been for tea at the Excelsior.
    The macarons des soeurs were great!
J:  Well, the benedictine nuns did know how to cook.
```

- Notice that some information is presupposed in this dialogue (as part of the background knowledge shared by Ivonne and Jean).
- If we change"macarons des soeurs" by "sushi" the dialogue loses all it sense.

Inference and NLP

Let's consider the following dialogue between Ivonne and Jean:

```
I:  I've just been for tea at the Excelsior.
    The macarons des soeurs were great!
J:  Well, the benedictine nuns did know how to cook.
```

- Notice that some information is presupposed in this dialogue (as part of the background knowledge shared by Ivonne and Jean).
- If we change"macarons des soeurs" by "sushi" the dialogue loses all it sense.
- What gives "integrity" to the dialogue are inference chains that Ivonne and Jean build.

Inference and NLP

Let's consider the following dialogue between Ivonne and Jean:

```
I:  I've just been for tea at the Excelsior.
    The macarons des soeurs were great!
J:  Well, the benedictine nuns did know how to cook.
```

- Notice that some information is presupposed in this dialogue (as part of the background knowledge shared by Ivonne and Jean).
- If we change"macarons des soeurs" by "sushi" the dialogue loses all it sense.
- What gives "integrity" to the dialogue are inference chains that Ivonne and Jean build.
- An important amount of information is left implicit in the dialogue (and it would be unnatural to make it explicit).

NLP and Inference

NLP and Inference

```
I:   I've just been for tea at the Excelsior.
     The macarons des soeurs were great.
J:   The benedictine nuns did know how to cook!
```

NLP and Inference

```
I:   I've just been for tea at the Excelsior.
     The macarons des soeurs were great.
J:   The benedictine nuns did know how to cook!
```

A program that aims to process/produce dialogs like the one in the example, needs (among others):

- transform the natural language into some kind of formal representation,

NLP and Inference

```
I:   I've just been for tea at the Excelsior.
     The macarons des soeurs were great.
J:   The benedictine nuns did know how to cook!
```

A program that aims to process/produce dialogs like the one in the example, needs (among others):

- transform the natural language into some kind of formal representation,
- be able to perform inference tasks like the ones we mentioned, over such representation

NLP and Inference

```
I:  I've just been for tea at the Excelsior.
    The macarons des soeurs were great.
J:  The benedictine nuns did know how to cook!
```

A program that aims to process/produce dialogs like the one in the example, needs (among others):

- transform the natural language into some kind of formal representation,
- be able to perform inference tasks like the ones we mentioned, over such representation
- and being able to handle an important amount of world or background knowledge.

Inference in NLP: Detailed Examples

Inference in NLP: Detailed Examples

- As we intuitively discussed, certain inference tasks are an essential part of natural language processing.

Inference in NLP: Detailed Examples

- As we intuitively discussed, certain inference tasks are an essential part of natural language processing.
- Notions like inference and consistency are used (in a crucial way) in tasks like:
  - To verify that a given enunciate is adequate in a given situation.

Inference in NLP: Detailed Examples

- As we intuitively discussed, certain inference tasks are an essential part of natural language processing.
- Notions like inference and consistency are used (in a crucial way) in tasks like:
  - To verify that a given enunciate is adequate in a given situation.
  - Resolution of nominal expressions.

Inference in NLP: Detailed Examples

- As we intuitively discussed, certain inference tasks are an essential part of natural language processing.
- Notions like inference and consistency are used (in a crucial way) in tasks like:
  - To verify that a given enunciate is adequate in a given situation.
  - Resolution of nominal expressions.
  - Explicitation of the implicit assumptions in presuppositions.

Inference in NLP: Detailed Examples

- As we intuitively discussed, certain inference tasks are an essential part of natural language processing.
- Notions like inference and consistency are used (in a crucial way) in tasks like:
  - To verify that a given enunciate is adequate in a given situation.
  - Resolution of nominal expressions.
  - Explicitation of the implicit assumptions in presuppositions.
  - Scope analysis.

Inference in NLP: Detailed Examples

- As we intuitively discussed, certain inference tasks are an essential part of natural language processing.
- Notions like inference and consistency are used (in a crucial way) in tasks like:
  - To verify that a given enunciate is adequate in a given situation.
  - Resolution of nominal expressions.
  - Explicitation of the implicit assumptions in presuppositions.
  - Scope analysis.
  - Ellipsis.

Ex. 1: Informativity and Consistency

Ex. 1: Informativity and Consistency

- As we say in the example with Ivonne and Jean, a dialogue and in general any natural language text, has rules that governs its structure.

Ex. 1: Informativity and Consistency

- As we say in the example with Ivonne and Jean, a dialogue and in general any natural language text, has rules that governs its structure.
- To desobey such rules makes the text: less natural, or suggest a particular reading, or appropriate only in particular situations, or (when everything else fails) meaningless.

Ex. 1: Informativity and Consistency

- As we say in the example with Ivonne and Jean, a dialogue and in general any natural language text, has rules that governs its structure.
- To desobey such rules makes the text: less natural, or suggest a particular reading, or appropriate only in particular situations, or (when everything else fails) meaningless.
- Two basic conditions are informativity and consistency.

Ex. 1: Informativity and Consistency

- As we say in the example with Ivonne and Jean, a dialogue and in general any natural language text, has rules that governs its structure.
- To desobey such rules makes the text: less natural, or suggest a particular reading, or appropriate only in particular situations, or (when everything else fails) meaningless.
- Two basic conditions are informativity and consistency.
- This conditions can be automatically verified using inference systems.

Ex. 1: Informativity and Consistency

Ex. 1: Informativity and Consistency

Informativity: A sentence should be informative with respect to the previous discourse, i.e., it should not be redundant and it should add information to the discourse.

```
Mary is a mother.
She has a child.
```

Ex. 1: Informativity and Consistency

Informativity: A sentence should be informative with respect to the previous discourse, i.e., it should not be redundant and it should add information to the discourse.

```
Mary is a mother.
She has a child.
```

The second sentence can be inferred from the first (given suitable background knowledge):

$$\frac{\begin{array}{ll} \mathit{Mother}(\mathit{Mary}) & \\ \forall x.(\mathit{Mother}(x) \rightarrow \exists y.\mathit{ChildOf}(x,y)) & \text{(Backgr. Information } (\star)) \end{array}}{\exists y.\mathit{ChildOf}(\mathit{Mary},y))}$$

Ex. 1: Informativity and Consistency

Ex. 1: Informativity and Consistency

- It is possible to automatically built a semantic representation φ of a phrase in natural language.

Ex. 1: Informativity and Consistency

- It is possible to automatically built a semantic representation φ of a phrase in natural language.
- Assuming background information Γ (as in $(\star)$) we can verify that φ is informative with respect to the previous discourse Δ, verifying that $\Delta \cup \Gamma \not\models \varphi$.

Ex. 1: Informativity and Consistency

Ex. 1: Informativity and Consistency

Consistency: A sentence should be consistent with the previous discourse, i.e., It should not contradict previous information.

Ex. 1: Informativity and Consistency

Consistency: A sentence should be consistent with the previous discourse, i.e., It should not contradict previous information.

```
Mary is a mother.
She doesn't have a child.
```

In this case the second sentence is not consistent with the information given in the previous discourse.

Ex. 1: Informativity and Consistency

Consistency: A sentence should be consistent with the previous discourse, i.e., It should not contradict previous information.

```
Mary is a mother.
She doesn't have a child.
```

In this case the second sentence is not consistent with the information given in the previous discourse.

- Assuming that Δ represents the previous discourse, Γ the world knowledge (and that $\Delta \cup \Gamma$ are consistent), and φ the sentence we want to check for consistency,

Ex. 1: Informativity and Consistency

Consistency: A sentence should be consistent with the previous discourse, i.e., It should not contradict previous information.

```
Mary is a mother.
She doesn't have a child.
```

In this case the second sentence is not consistent with the information given in the previous discourse.

- Assuming that Δ represents the previous discourse, Γ the world knowledge (and that $\Delta \cup \Gamma$ are consistent), and φ the sentence we want to check for consistency,
- we can use an inference system called "model builder" to try to build a model of $\Delta \cup \Gamma \cup \{\varphi\}$.

Ex. 1: Informativity and Consistency

- Notice that the test of informativity and consistency are related but they are both necessary.

Ex. 1: Informativity and Consistency

- Notice that the test of informativity and consistency are related but they are both necessary.
- A sentence can be:
  - Informative but not consistent,
  - Consistent but not informative, or
  - Both.

Ex. 1: Informativity and Consistency

- Notice that the test of informativity and consistency are related but they are both necessary.
- A sentence can be:
  - Informative but not consistent,
  - Consistent but not informative, or
  - Both.
  - (Can it be neither?)

Ex. 1: Informativity and Consistency

Ex. 1: Informativity and Consistency

- **Note:** There are (many!) other notions that govern the corrext structure of a discourse. For example, a sentence usually has to be relevant with respect to the previous discourse.

```
Mary is a mother.
The sky is blue.
```

Ex. 1: Informativity and Consistency

- **Note:** There are (many!) other notions that govern the correxct structure of a discourse. For example, a sentence usually has to be relevant with respect to the previous discourse.

```
Mary is a mother.
The sky is blue.
```

- The second sentence is informative and consistent with respect to the previous discourse, but in any case the discourse seems incoherent. The problem being that the two phrases seems unrelated. The contribution of the second phrase doesn't seem relevant to the discourse.

Ex. 1: Informativity and Consistency

- **Note:** There are (many!) other notions that govern the correxct structure of a discourse. For example, a sentence usually has to be relevant with respect to the previous discourse.

```
Mary is a mother.
The sky is blue.
```

- The second sentence is informative and consistent with respect to the previous discourse, but in any case the discourse seems incoherent. The problem being that the two phrases seems unrelated. The contribution of the second phrase doesn't seem relevant to the discourse.
- Checking relevance (or even formally defining when a given sentence is relevant) is a non trivial problem.

Ex. 2: Resolving Nominal Expressions

Ex. 2: Resolving Nominal Expressions

- During a discourse referents (the objects and subjects mentioned) are introduced.

Ex. 2: Resolving Nominal Expressions

- During a discourse referents (the objects and subjects mentioned) are introduced.
  When these referents are mentioned afterwards, usually a nominal expression will be used (e.g., pronouns) which usually will let us univocably pick the referent.

```
John and Mary went to the cinema.
He would have liked to go to the bar.
```

Ex. 2: Resolving Nominal Expressions

- During a discourse referents (the objects and subjects mentioned) are introduced.
  When these referents are mentioned afterwards, usually a nominal expression will be used (e.g., pronouns) which usually will let us univocably pick the referent.

  ```
  John and Mary went to the cinema.
  He would have liked to go to the bar.
  ```

- This sometimes combines with the introduction of new referents, and the nominal expression might require fairly involved inference in terms of background knowledge to be resolved.

  ```
  John and Mary went to the cinema.
  Their daughter would meet them in the main hall.
  ```

Ex. 3: Presuppositions

Ex. 3: Presuppositions

- Many expressions involved presuppositions that govern the adequacy of their use. The sentence

  `Open the door.`

  has as a presupposition that the door is actually closed (in this case, this is a presupposition of the verb `to open`).

Ex. 3: Presuppositions

- Many expressions involved presuppositions that govern the adequacy of their use. The sentence

  `Open the door.`

  has as a presupposition that the door is actually closed (in this case, this is a presupposition of the verb `to open`).
- The fulfillment of the presuppositions of a given word can strongly impact the adequateness of the use of a given world. E.g.: also

```
1.    Mary likes to swim.
2a.   John also loves water sports.
```

Ex. 3: Presuppositions

- Many expressions involved presuppositions that govern the adequacy of their use. The sentence

  `Open the door.`

  has as a presupposition that the door is actually closed (in this case, this is a presupposition of the verb `to open`).
- The fulfillment of the presuppositions of a given word can strongly impact the adequateness of the use of a given world. E.g.: also

```
1.    Mary likes to swim.
2a.   John also loves water sports.
2b.   John also loves riding a horse.
```

Ex. 3: Presuppositions

- Many expressions involved presuppositions that govern the adequacy of their use. The sentence

  `Open the door.`

  has as a presupposition that the door is actually closed (in this case, this is a presupposition of the verb `to open`).
- The fulfillment of the presuppositions of a given word can strongly impact the adequateness of the use of a given world. E.g.: also

```
1.    Mary likes to swim.
2a.   John also loves water sports.
2b.   John also loves riding a horse.
2c.   John also hates getting wet.
```

Ex. 4: Scope

Ex. 4: Scope

- Natural language is full of ambiguity, ranging from lexical ambiguity (e.g., bank). to complex cases having to do with the different scope possibilities of quantified phrases.

Ex. 4: Scope

- Natural language is full of ambiguity, ranging from lexical ambiguity (e.g., bank). to complex cases having to do with the different scope possibilities of quantified phrases.
- The phrase

  `Every man loves a woman.`

  has two different interpretations,

Ex. 4: Scope

- Natural language is full of ambiguity, ranging from lexical ambiguity (e.g., bank). to complex cases having to do with the different scope possibilities of quantified phrases.
- The phrase

  `Every man loves a woman.`

  has two different interpretations, one where each man has one (possible different) loved woman,
  $\forall x.(Man(x) \rightarrow \exists y.(Woman(y) \wedge Loves(x, y))))$

Ex. 4: Scope

- Natural language is full of ambiguity, ranging from lexical ambiguity (e.g., bank). to complex cases having to do with the different scope possibilities of quantified phrases.
- The phrase

  `Every man loves a woman.`

  has two different interpretations, one where each man has one (possible different) loved woman, $\forall x.(Man(x) \rightarrow \exists y.(Woman(y) \wedge Loves(x, y))))$ and another where every man love the same woman (E.g. Sofia Loren) $(\exists y.\forall x.(Man(x) \rightarrow (Woman(y) \wedge Loves(x, y))))$.

Ex. 4: Scope

- Natural language is full of ambiguity, ranging from lexical ambiguity (e.g., bank). to complex cases having to do with the different scope possibilities of quantified phrases.
- The phrase

  `Every man loves a woman.`

  has two different interpretations, one where each man has one (possible different) loved woman, $\forall x.(Man(x) \rightarrow \exists y.(Woman(y) \wedge Loves(x, y))))$ and another where every man love the same woman (E.g. Sofia Loren) $(\exists y.\forall x.(Man(x) \rightarrow (Woman(y) \wedge Loves(x, y))))$.
- But in the following example

  `Every boxer has a broken nose.`

  only one of the interpretations is possible!.

Ex. 5: Ellipsis

Ex. 5: Ellipsis

- Another common tool in natural language to avoid repetitions (but which instead might be a source of ambiguities) is the use of parallel structures (e.g., ellipsis).

  `John wants cake, and Ed too.`

Ex. 5: Ellipsis

- Another common tool in natural language to avoid repetitions (but which instead might be a source of ambiguities) is the use of parallel structures (e.g., ellipsis).

  `John wants cake, and Ed too.`

- The following example

  `John takes care of his car, and Ed too.`

Ex. 5: Ellipsis

- Another common tool in natural language to avoid repetitions (but which instead might be a source of ambiguities) is the use of parallel structures (e.g., ellipsis).

  `John wants cake, and Ed too.`

- The following example

  `John takes care of his car, and Ed too.`

  has two interpretation, depending on which car Ed is taking care of (his own or John's).

Ex. 5: Ellipsis

- Another common tool in natural language to avoid repetitions (but which instead might be a source of ambiguities) is the use of parallel structures (e.g., ellipsis).

  `John wants cake, and Ed too.`

- The following example

  `John takes care of his car, and Ed too.`

  has two interpretation, depending on which car Ed is taking care of (his own or John's).

- Notice that understand the correct interpretation can sometimes crucial :-)

  `John is in love with his wife, and Ed too.`

A Bit of History

A Bit of History

During the 70s, investigations into the use of inference in NLP was mainstream.

A Bit of History

During the 70s, investigations into the use of inference in NLP was mainstream.

- One of the main examples was Roger Schank's PhD thesis on tale understanding: R. Schank, "Conceptual Dependency: A Theory of Natural Language Understanding," *Cognitive Psychology*, 1972.)

A Bit of History

During the 70s, investigations into the use of inference in NLP was mainstream.

- One of the main examples was Roger Schank's PhD thesis on tale understanding: R. Schank, "Conceptual Dependency: A Theory of Natural Language Understanding," *Cognitive Psychology*, 1972.)
- But the main outcome of this work was negative:
  The amount of information necessary and the complexity of the reasoning tasks involved to "understand" a text are just too demanding for the existing inference systems.

A Bit of History

During the 70s, investigations into the use of inference in NLP was mainstream.

- One of the main examples was Roger Schank's PhD thesis on tale understanding: R. Schank, "Conceptual Dependency: A Theory of Natural Language Understanding," *Cognitive Psychology*, 1972.)
- But the main outcome of this work was negative:
  The amount of information necessary and the complexity of the reasoning tasks involved to "understand" a text are just too demanding for the existing inference systems.
- From then on, much of the work in NLP shifted to more tractable topics, like morphology analysis, grammar design and syntactic processing.

Nowadays

Nowadays

Things have drastically change in the last decade:

Nowadays

Things have drastically change in the last decade:

- There are much more powerful inference systems than in the 70s, able to perform inference in terms of ontologies containing thousands of definitions.

Nowadays

Things have drastically change in the last decade:

- There are much more powerful inference systems than in the 70s, able to perform inference in terms of ontologies containing thousands of definitions.
- There are extensive on-line repositories of lexical information (WordNet, FrameNet, etc). As this is lexical information, it is fairly independent of a particular domain, and can be used in diverse applications.

Nowadays

Things have drastically change in the last decade:

- There are much more powerful inference systems than in the 70s, able to perform inference in terms of ontologies containing thousands of definitions.
- There are extensive on-line repositories of lexical information (WordNet, FrameNet, etc). As this is lexical information, it is fairly independent of a particular domain, and can be used in diverse applications.
- There are new applications that require NLP (question answering, information extraction, text summarisation), that require inference tasks less demanding than the "full text comprehension" studied by Schank.

Exercise

- Use DP to prove that the following graph is not 2 colorable.