CMSC 471:
Some AI Techniques in Machine Learning

Slides courtesy Frank Ferraro

Outline

- Structured Prediction
- Example Applications
- Some Examples of Search
- Some Examples of Logic

An Extension of Constraints: Integer Linear Programming

Classification

Classification: provide *labels* to an input item

Labels are application/task dependent

Machine learning classification: Learn a function p_α to provide these labels automatically

Classification

Classification: provide *labels* to an input item

Labels are application/task dependent

Machine learning classification: Learn a function p_α to provide these labels automatically

We assume there are some "weights" (parameters) that control the behavior of p_α.

Classification

Classification: provide *labels* to an input item

Labels are application/task dependent

Machine learning classification: Learn a function p_α to provide these labels automatically

Input:

Electronic alerts have been used to assist the authorities in moments of chaos and potential danger: after the Boston bombing in 2013, when the Boston suspects were still at large, and last month in Los Angeles, during an active shooter scare at the airport.

Possible output labels:

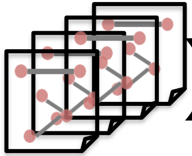
TECH

NOT TECH

Classification vs. Structured Prediction

“Flat” prediction

y is a single label

$y = p_{\alpha}($  $)$

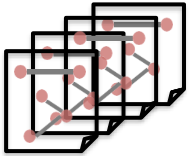
Examples:

- Document classification
  - Label a doc with its “topic”
- Image classification
  - E.g., identify the (main) item in an image
- Robot action prediction
  - Determine what action a robot should take

Classification vs. Structured Prediction

“Flat” prediction

y is a single label

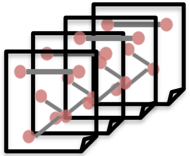
$y = p_{\boldsymbol{\alpha}}($  $)$

Examples:

- Document classification
  - Label a doc with its “topic”
- Image classification
  - E.g., identify the (main) item in an image
- Robot action prediction
  - Determine what action a robot should take

Structured prediction

y has some internal structure to predict

$(y_1, y_2, \ldots, y_M) = p_{\boldsymbol{\alpha}}($  $)$

Examples:

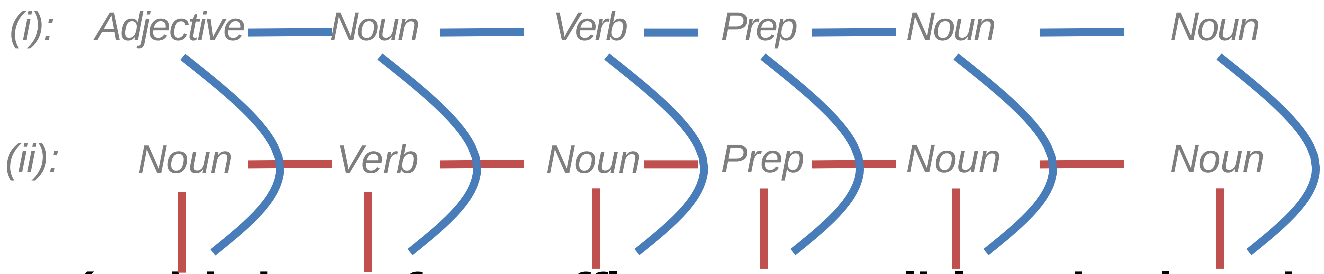
- Part of speech tagging
  - Identify each word in a sentence as a noun, verb, etc.
- Action identification in video

Example: Part-of-Speech Sequence Tagging

(i): Adjective — Noun — Verb — Prep — Noun — Noun

p(British Left Waffles on Falkland Islands)

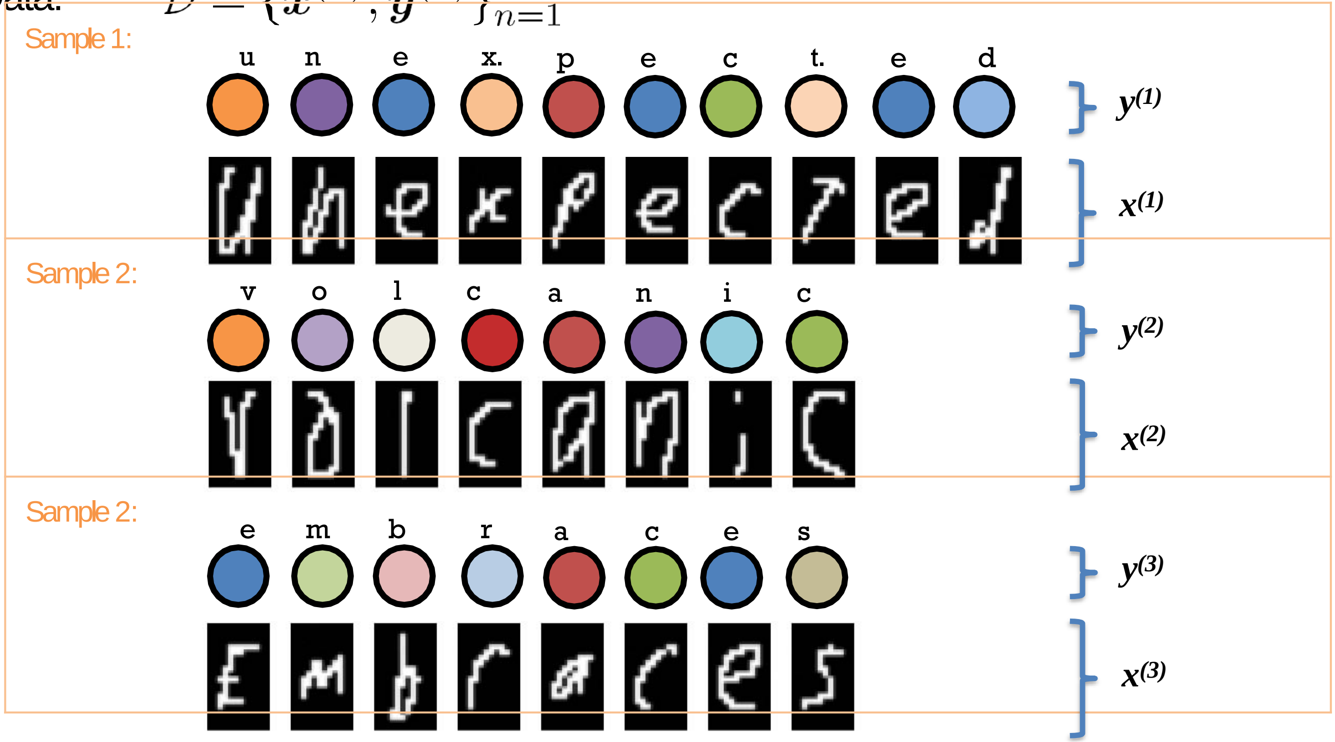
Example: Part-of-Speech Sequence Tagging



p(British Left Waffles on Falkland Islands)

Example: Handwriting Recognition

Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$



Figures from (Chatzis & Demiris, 2013)

Alignment

Le chat est sur la chaise.

The cat is on the chair.

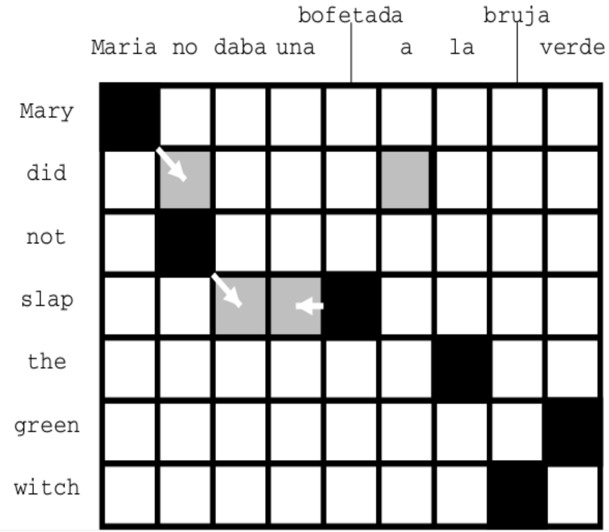
?

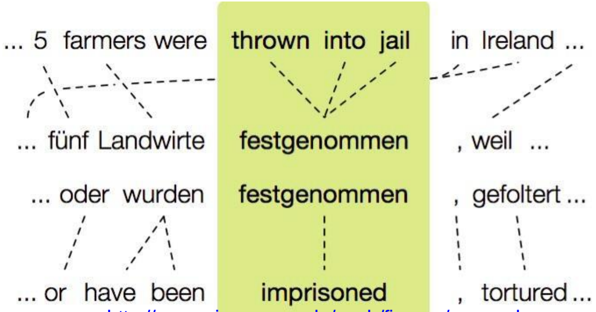
Le chat est sur la chaise.

The cat is on the chair.

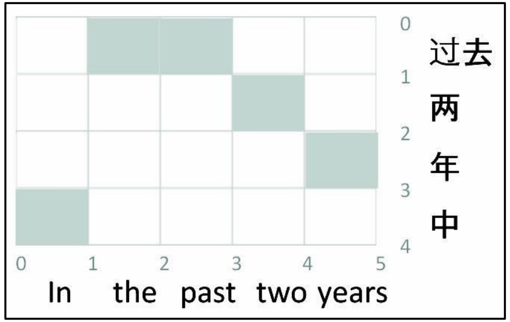
$$p(English|French) \propto p(French|English) * p(English)$$

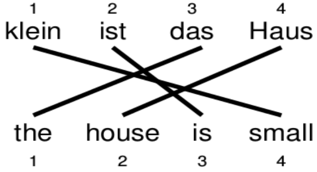
Example: Word Alignment, Phrase Extraction





http://www.cis.upenn.edu/~ccb/figures/research
-statement/pivoting.jpg





Example: Object Recognition

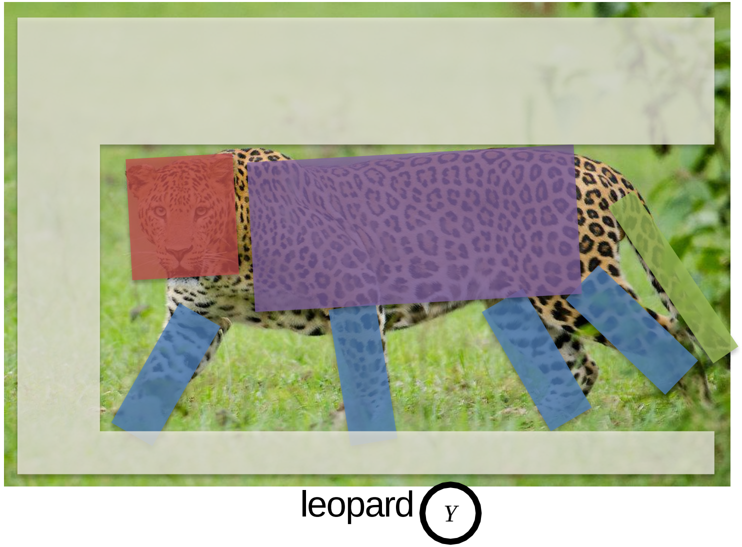
Data consists of images x and labels y.

leopard Y

Example: Object Recognition

Data consists of images x and labels y.

Preprocess data into "patches"

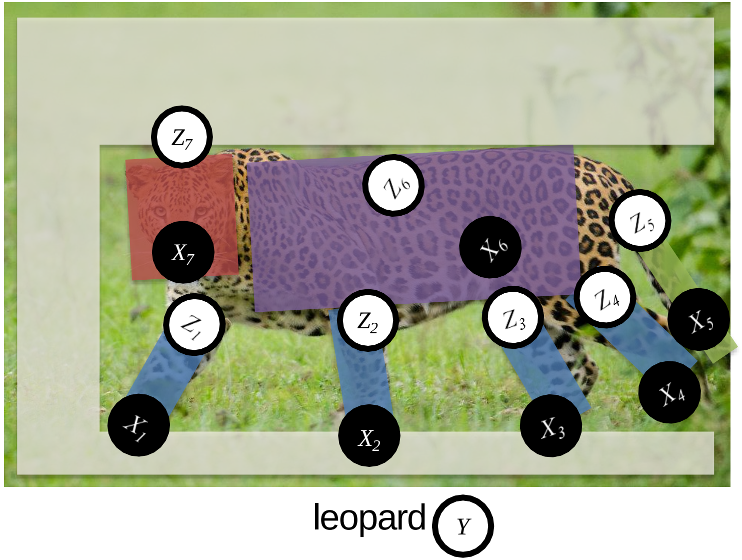


Example: Object Recognition

Data consists of images x and labels y.

Preprocess data into "patches"

Posit a latent labeling z describing the object's parts (e.g. head, leg, tail, torso, grass)



Example: Object Recognition

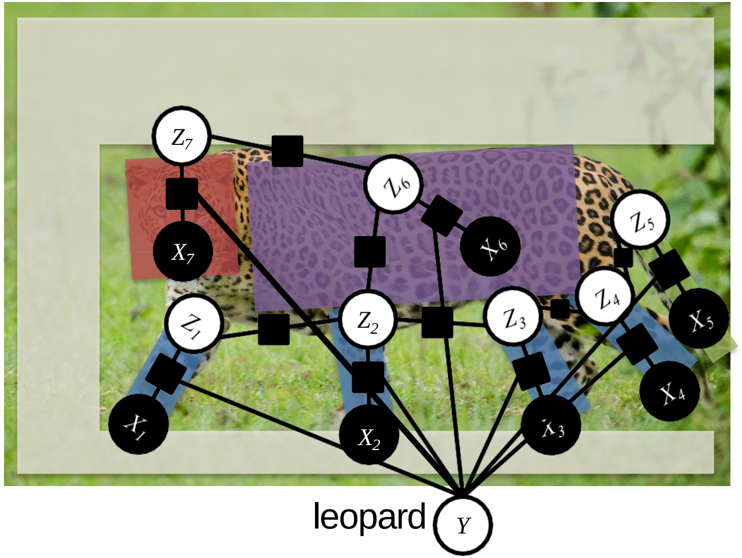
Data consists of images x and labels y.

Preprocess data into "patches"

Posit a latent labeling z describing the object's parts (e.g. head, leg, tail, torso, grass)

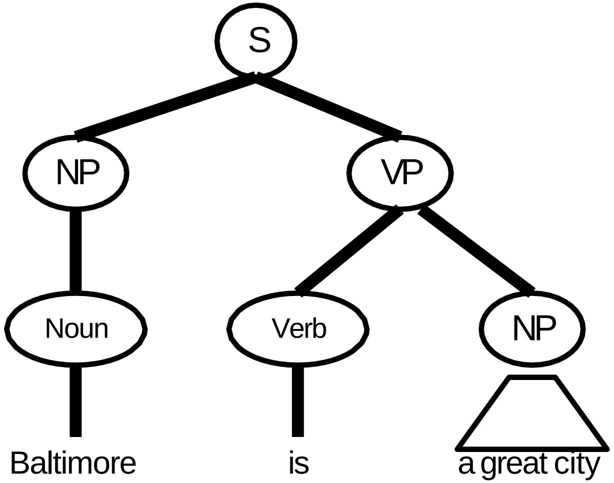
Define a "**graphical model**" with these latent variables in mind

z is not observed at train or test time



Example: Sentence Parsing

Baltimore is a great city



Sentence

Structured analysis (**parse**) of the sentence (diagramming a sentence)

Outline

- Structured Prediction
- Example Applications
- Some Examples of Search
- Some Examples of Logic

An Extension of Constraints: Integer Linear Programming

Search in NLP

An Efficient **A* Search** Algorithm for Statistical Machine Translation
www.aclweb.org › anthology
File Format: PDF/Adobe Acrobat
on Computational Linguistics, pages 1086–1090, Saarbrücken, Germany, August . F. J. Och and H. Ney. 2000b. Improved statistical alignment models. In Proc. of ...

Algorithms for an Optimal **A* Search** and Linearizing the **Search** in ...
www.aclweb.org › anthology
File Format: PDF/Adobe Acrobat
Abstract. The stack decoder is an attractive algorithm for con- trolling the acoustic and language model matching in a continuous speech recognizer.

Multi-Document Summarization Using **A* Search** and Discriminative ...
ACL Anthology › anthology
Anthology ID: D10-1047; Volume: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing; Month: October; Year: 2010 ...

A Beam-Search Decoder for Grammatical Error Correction - ACL ...
ACL Anthology › anthology
Anthology ID: D12-1052; Volume: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural ...

Approximate Factoring for **A* Search** - ACL Anthology
ACL Anthology › anthology
Anthology ID: N07-1052; Volume: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational ...

Learning Architectures from **an Extended Search** Space for ...
ACL Anthology › anthology › 2020.acl-main.592
We implement our model in a differentiable architecture **search** system. For recurrent neural language modeling, it outperforms a strong baseline significantly on ...

BERT-kNN: Adding **a kNN Search** Component to Pretrained ...

https://www.aclweb.org/anthology/search/?q=a*+search

An Efficient A* Search Algorithm for Statistical Machine Translation

Franz Josef Och, Nicola Ueffing, Hermann Ney

Lehrstuhl für Informatik VI, Computer Science Department
RWTH Aachen - University of Technology
D-52056 Aachen, Germany
{och,ueffing,ney}@informatik.rwth-aachen.de

Abstract

In this paper, we describe an efficient A* search algorithm for statistical machine translation. In contrary to beam-search or greedy approaches it is possible to guarantee the avoidance of search errors with A*. We develop various sophisticated admissible and almost admissible heuristic functions. Especially our newly developped method to perform a multi-pass A* search with an iteratively improved heuristic function allows us to translate even long sentences. We compare the A* search algorithm with a beam-search approach on the Hansards task.

1 Introduction

The goal of machine translation is the translation of a text given in some source language into a target language. We are given a source string $f_1^J = f_1...f_j...f_J$, which is to be translated into a target string $e_1^I = e_1...e_i...e_I$. Among all possible target strings, we will choose the string with the highest probability:

$$\begin{aligned} \hat{e}_1^I &= \arg\max_{e_1^I} \left\{ Pr(e_1^I|f_1^J) \right\} \\ &= \arg\max_{e_1^I} \left\{ Pr(e_1^I) \cdot Pr(f_1^J|e_1^I) \right\} \end{aligned}$$

The argmax operation denotes the search problem, i.e. the generation of the output sentence in the target language. $Pr(e_1^I)$ is the **language model** of the target language, whereas $Pr(f_1^J|e_1^I)$ denotes the **translation model**.

Many statistical translation models (Brown et al., 1993; Vogel et al., 1996; Och and Ney, 2000b) try to model word-to-word correspondences between source and target words. These correspondences are called an **alignment**. The model is often further restricted in a way such that each source word is assigned *exactly one* target word. The alignment mapping is $j \to i = a_j$ from source position j to target position $i = a_j$. The alignment a_1^J may contain alignments $a_j = 0$ with the 'empty' word e_0 to account for source words that are not aligned to any target word. In (statistical) alignment models $Pr(f_1^J, a_1^J|e_1^I)$, the alignment a_1^J is introduced as a hidden variable.

Typically, the search is performed using the so-called maximum approximation:

$$\begin{aligned} \hat{e}_1^I &= \arg\max_{e_1^I} \left\{ Pr(e_1^I) \cdot \sum_{a_1^J} Pr(f_1^J, a_1^J|e_1^I) \right\} \\ &= \arg\max_{e_1^I} \left\{ Pr(e_1^I) \cdot \max_{a_1^J} Pr(f_1^J, a_1^J|e_1^I) \right\} \end{aligned}$$

The search space consists of the set of all possible target language strings e_1^I and all possible alignments a_1^J.

2 IBM Model 4

Various statistical alignment models of the form $Pr(f_1^J, a_1^J|e_1^I)$ have been introduced in (Brown et al., 1993; Vogel et al., 1996; Och and Ney, 2000a). In this paper we use the so-called Model 4 from (Brown et al., 1993).

In Model 4 the statistical alignment model is decomposed into five sub-models:

- the lexicon model $p(f|e)$ for the probability that the source word f is a translation of the target word e,
- the distortion model $p_{=1}(j-j'|C(f_j), E)$ for the probability that the translations of two

Core idea: searching for the correct translation

- State: the current partial translation
- Actions: translating the next word

UMBC

An Efficient A* Search Algorithm for Statistical Machine Translation

Franz Josef Och, Nicola Ueffing, Hermann Ney

Lehrstuhl für Informatik VI, Computer Science Department
RWTH Aachen - University of Technology
D-52056 Aachen, Germany
{och,ueffing,ney}@informatik.rwth-aachen.de

Abstract

In this paper, we describe an efficient A* search algorithm for statistical machine translation. In contrary to beam-search or greedy approaches it is possible to guarantee the avoidance of search errors with A*. We develop various sophisticated admissible and almost admissible heuristic functions. Especially our newly developped method to perform a multi-pass A* search with an iteratively improved heuristic function allows us to translate even long sentences. We compare the A* search algorithm with a beam-search approach on the Hansards task.

1 Introduction

The goal of machine translation is the translation of a text given in some source language into a target language. We are given a source string $f_1^J = f_1 ... f_j ... f_J$, which is to be translated into a target string $e_1^I = e_1 ... e_i ... e_I$. Among all possible target strings, we will choose the string with the highest probability:

$$\begin{aligned} \hat{e}_1^I &= \arg\max_{e_1^I} \left\{ Pr(e_1^J | f_1^I) \right\} \\ &= \arg\max_{e_1^I} \left\{ Pr(e_1^I) \cdot Pr(f_1^J | e_1^I) \right\} \end{aligned}$$

The argmax operation denotes the search problem, i.e. the generation of the output sentence in the target language. $Pr(e_1^I)$ is the **language model** of the target language, whereas $Pr(f_1^J | e_1^I)$ denotes the **translation model**.

Many statistical translation models (Brown et al., 1993; Vogel et al., 1996; Och and Ney, 2000b) try to model word-to-word correspondences between source and target words. These correspondences are called an **alignment**. The model is often further restricted in a way such that each source word is assigned *exactly one* target word. The alignment mapping is $j \to i = a_j$ from source position j to target position $i = a_j$. The alignment a_1^J may contain alignments $a_j = 0$ with the 'empty' word e_0 to account for source words that are not aligned to any target word. In (statistical) alignment models $Pr(f_1^J, a_1^J | e_1^I)$, the alignment a_1^J is introduced as a hidden variable.

Typically, the search is performed using the so-called maximum approximation:

$$\begin{aligned} \hat{e}_1^I &= \arg\max_{e_1^I} \left\{ Pr(e_1^I) \cdot \sum_{a_1^J} Pr(f_1^J, a_1^J | e_1^I) \right\} \\ &= \arg\max_{e_1^I} \left\{ Pr(e_1^I) \cdot \max_{a_1^J} Pr(f_1^J, a_1^J | e_1^I) \right\} \end{aligned}$$

The search space consists of the set of all possible target language strings e_1^I and all possible alignments a_1^J.

2 IBM Model 4

Various statistical alignment models of the form $Pr(f_1^J, a_1^J | e_1^I)$ have been introduced in (Brown et al., 1993; Vogel et al., 1996; Och and Ney, 2000a). In this paper we use the so-called Model 4 from (Brown et al., 1993).

In Model 4 the statistical alignment model is decomposed into five sub-models:

- the **lexicon model** $p(f|e)$ for the probability that the source word f is a translation of the target word e,
- the distortion model $p_{=1}(j - j' | C(f_j), E)$ for the probability that the translations of two

Core idea: searching for the correct translation

- State: the current partial translation
- Actions: translating the next word

Cool part: an admissible heuristic!

- "heuristic function including a coupling between translation, fertility, and language model probabilities [scores]"
- How do they show it? By using knowledge about how *their* particular approach computes these scores.

Logic in NLP

Augmenting Neural Networks with **First-order Logic** - ACL Anthology
ACL Anthology › anthology
We evaluate our modeling strategy on three tasks: machine comprehension, natural language inference, and text chunking. Our exp

Leveraging Declarative Knowledge in Text and **First-Order Logic** for ...
ACL Anthology › anthology › 2020.emnlp-main.320
Specifically, we leverage the declarative knowledge expressed in both **first-order logic** and natural language. The former refers to the logical consistency ...

Discourse Level Explanatory Relation Extraction from Product ...
www.aclweb.org › anthology
File Format: PDF/Adobe Acrobat
Oct 18, 2013 ... In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational. Natural Language ...

Question Answering over Linked Data Using **First-order Logic** - ACL ...
ACL Anthology › anthology
Anthology ID: D14-1116; Volume: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP); Month: October; Year: ...

Discourse Level Explanatory Relation Extraction from Product ...
ACL Anthology › anthology
Anthology ID: D13-1097; Volume: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing; Month: October; Year: 2013 ...

Expressing Disjunctive and Negative Feature Constraints With ...
ACL Anthology › anthology
Expressing Disjunctive and Negative Feature Constraints With Classical **First- Order Logic**. Mark Johnson. Anthology ID: P90-1022; Volume: 28th Annual ...

https://www.aclweb.org/anthology/search/?q=first+order+logic

UMBC

Knowledge Graph Inference using Tensor Embedding

Ankur Padia[1] *, Konstantinos Kalpakis[2], Francis Ferraro[2], Tim Finin[2]

[1]Philips Research North Americas, Cambridge, MA 02141, USA

[2]University of Maryland, Baltimore County, Baltimore, MD 21250, USA

ankur.padia@philips.com, {kalpakis, ferraro, finin}@umbc.edu

1 Introduction

This extended abstract describes work reported in the Journal of Web Semantics (Padia et al. 2019).

Axiom based inference provides a clear and consistent way of reasoning to add more information to a knowledge graph. However, constructing a set of axioms is expensive and requires domain expertise, time, and money. It is also difficult to reuse or adapt a set of axioms to a knowledge graph in a new domain or even in the same domain but using a slightly different representation approach.

Representation learning (Bengio, Courville, and Vincent 2013) introduces a way to augment or even replace manually constructed ontology axioms and rules by using knowledge graph instances to discover common patterns and then apply them to suggest changes to the graph. The changes are often in the form of adding missing types and relations, but can also include schema modifications, removing incoherent instances, merging sets of instances describing the same real-world entity, or adding relation probabilities.

One popular approach for representation learning is based on learning how to embed a graph's entities and relations into a real-valued vector space, allowing both to be represented by dense, real-valued vectors. The entity and relation embeddings can be learned either independently or jointly, and then used to predict additional relations that are missing. Jointly learning the embeddings allows each to enhance the other (Nickel, Tresp, and Kriegel 2011).

There are several models that learn embedding of entities and relations to perform inference. Some use existing schemas (Krompass, Baier, and Tresp 2015) to regularize the quality of the embedding. These often give better performance compared to those that do not use schemas, as in Nickel (2011). However, schema-based embedding methods suffer from the above-mentioned limitations. We have developed a family of novel methods that improves the quality of the embeddings without using pre-defined schemas (Padia et al. 2019; Padia 2019a).

We divide statistical models that infer additional knowl- *prediction or classification* systems that determine if a new fact holds or not. There are several approaches to link ranking (Socher et al. 2013), all of which involve an auxiliary problem of determining a threshold, either globally or per relation, that separates plausible from implausible relations. Since we are only interested in extending a knowledge graph with relations that are likely to hold (what we call facts), we designed an approach to solve it directly. Thus we have the fact or link prediction task: given a knowledge graph, learn a model that can classify relation instances that are very likely to hold. This task is more specific than link ranking and more directly solves an important problem.

We improve the quality of the relation and entities representations using data-driven constraints, hence our approach can be used when ontological axioms are not available or are expensive to create. We measured the quality of the learned embeddings by comparing our approaches with previous non-schema based methods as well as with neural models and found improvement ranging from 5% to 50%. We demonstrated its broad applicability using eight real-world data sets covering human language, medical data, and general world knowledge that are available online (Padia 2019b). This work makes three main contributions: it (1) provides a family of representation learning algorithms and an extensive analysis on eight datasets; (2) yields better results than existing tensor and neural models; and (3) includes a provably convergent factorization algorithm.

We use the initial knowledge graph to pre-compute a similarity matrix C for the relations that will help constrain the learning of embeddings. To better understand the idea, consider the WordNet knowledge graph where entities are words that are connected with relations like *hypernym* and *similar*. The graph has no schema and there are many missing relations. We create a similarity matrix quantifying the similarity between two relations as the number of overlapping words. The cells in Figure 1 show the number of subjects or objects that are shared by the relations, with darker cells indicating a smaller overlap.

…and local chances to get involved

(this combines neural networks, propositional logic-based knowledge bases, and structured prediction)

Outline

- Structured Prediction
- Example Applications
- Some Examples of Search
- Some Examples of Logic

An Extension of Constraints: Integer Linear Programming

Linear Programming

$$AW \leq B$$

$$a_1 w_1 + a_2 w_2 + \cdots a_M w_M \leq b$$

input data

$\max_W W\, x$ subject to linear constraints on W

matrix (or vector) of learned parameters

Example: The diet problem

A student wants to spend as little money on food while getting sufficient amount of vitamin Z and nutrient X. Her options are:

Item	Cost/100g	Vitamin Z	Nutrient X
Carrots	2	4	0.4
Sunflower seeds	6	10	4
Double cheeseburger	0.3	0.01	2

How should she spend her money to get at least 5 units of vitamin Z and 3 units of nutrient X?

Let c, s and d denote how much of each item is purchased

Minimize total cost

such that

At least 5 units of vitamin Z,

At least 3 units of nutrient X,

The number of units purchased is not negative

Example: The diet problem

A student wants to spend as little money on food while getting sufficient amount of vitamin Z and nutrient X. Her options are:

Item	Cost/100g	Vitamin Z	Nutrient X
Carrots	2	4	0.4
Sunflower seeds	6	10	4
Double cheeseburger	0.3	0.01	2

How should she spend her money to get at least 5 units of vitamin Z and 3 units of nutrient X?

Let c, s and d denote how much of each item is purchased

$$\min \quad 2c + 6s + 0.3d$$

Minimize total cost

such that

At least 5 units of vitamin Z,

At least 3 units of nutrient X,

The number of units purchased is not negative

Example: The diet problem

A student wants to spend as little money on food while getting sufficient amount of vitamin Z and nutrient X. Her options are:

Item	Cost/100g	Vitamin Z	Nutrient X
Carrots	2	4	0.4
Sunflower seeds	6	10	4
Double cheeseburger	0.3	0.01	2

How should she spend her money to get at least 5 units of vitamin Z and 3 units of nutrient X?

Let c, s and d denote how much of each item is purchased

$$\min \quad 2c + 6s + 0.3d$$

Minimize total cost

such that

$$4c + 10s + 0.01d \geq 5$$

At least 5 units of vitamin Z,

At least 3 units of nutrient X,

The number of units purchased is not negative

Example: The diet problem

A student wants to spend as little money on food while getting sufficient amount of vitamin Z and nutrient X. Her options are:

Item	Cost/100g	Vitamin Z	Nutrient X
Carrots	2	4	0.4
Sunflower seeds	6	10	4
Double cheeseburger	0.3	0.01	2

How should she spend her money to get at least 5 units of vitamin Z and 3 units of nutrient X?

Let c, s and d denote how much of each item is purchased

$$\min \quad 2c + 6s + 0.3d$$

Minimize total cost

such that

$$4c + 10s + 0.01d \geq 5$$

At least 5 units of vitamin Z,

$$0.4c + 4s + 2d \geq 3$$

At least 3 units of nutrient X,

The number of units purchased is not negative

Example: The diet problem

A student wants to spend as little money on food while getting sufficient amount of vitamin Z and nutrient X. Her options are:

Item	Cost/100g	Vitamin Z	Nutrient X
Carrots	2	4	0.4
Sunflower seeds	6	10	4
Double cheeseburger	0.3	0.01	2

How should she spend her money to get at least 5 units of vitamin Z and 3 units of nutrient X?

Let c, s and d denote how much of each item is purchased

$$\min \quad 2c + 6s + 0.3d$$

such that

$$4c + 10s + 0.01d \geq 5$$
$$0.4c + 4s + 2d \geq 3$$
$$c \geq 0, s \geq 0, d \geq 0.$$

Minimize total cost

At least 5 units of vitamin Z,

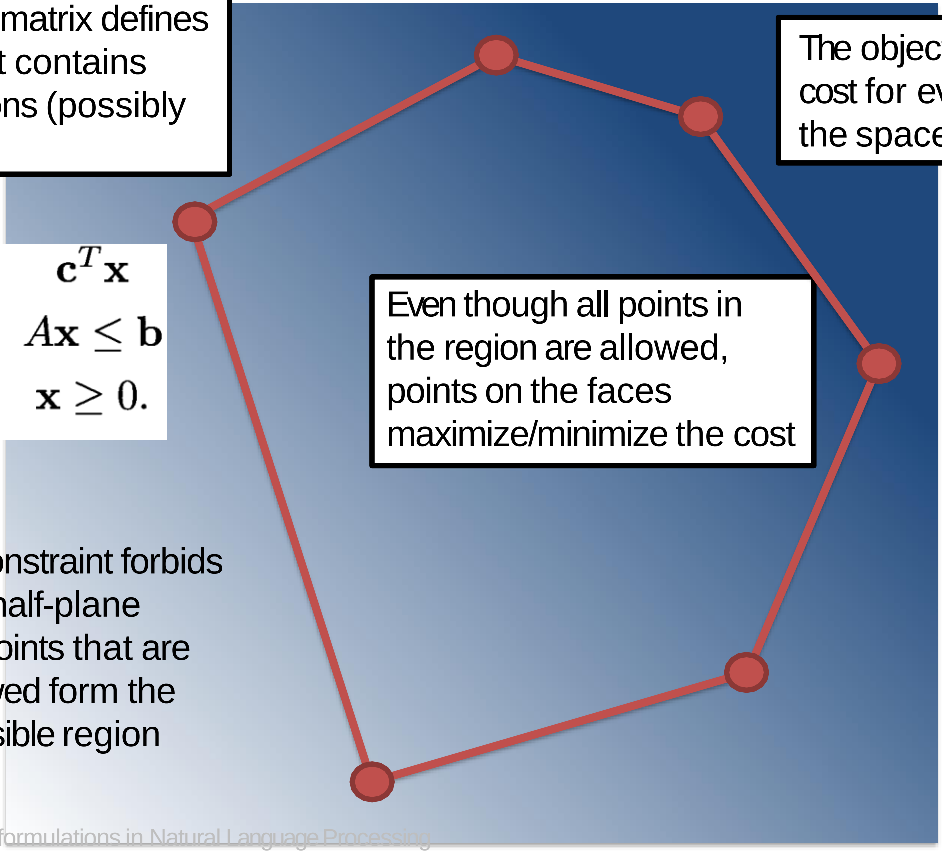
At least 3 units of nutrient X,

The number of units purchased is not negative

Geometric Views of the Constraints

The constraint matrix defines a polytope that contains allowed solutions (possibly not closed)

The objective defines cost for every point in the space



$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0. \end{aligned}$$

Even though all points in the region are allowed, points on the faces maximize/minimize the cost

Every constraint forbids a half-plane
The points that are allowed form the feasible region

Integer Linear Programming

Variables to assign

$$\max_{z} c^T z \text{ subject to}$$

c: a matrix (or vector) of learned parameters

1. linear constraints on z
2. Each z_k is an integer

Integer Linear Programming

$$Az \le B$$

$$a_1 z_1 + a_2 z_2 + \cdots a_M z_M \le b$$

Variables
to assign

$$\max_{z} c^T z \text{ subject to}$$

1. linear constraints on z
2. Each z_k is an integer

c: a matrix (or vector)
of learned parameters

Integer Linear Programming

$$Az \leq B$$

$$a_1 z_1 + a_2 z_2 + \cdots a_M z_M \leq b$$

Variables
to assign

$$\max_{z} c^T z \text{ subject to}$$

1. linear constraints on z
2. Each z_k is an integer

c: a matrix (or vector)
of learned parameters

Integer Linear Programming

$$Az \leq B$$

$$a_1 z_1 + a_2 z_2 + \cdots a_M z_M \leq b$$

Variables
to assign

$$\max_{z} c^T z \text{ subject to}$$

c: a matrix (or vector)
of learned parameters

1. linear constraints on z
2. Each z_k is an integer

Integer Linear Programming

Variables to assign

$$\max_{z} c^T z \text{ subject to}$$

c: a matrix (or vector) of learned parameters

$Az \le B$

$a_1 z_1 + a_2 z_2 + \cdots a_M z_M \le b$

1. linear constraints on z
2. Each z_k is an integer

ILP is NP-complete ☹

Integer Linear Programming

$$Az \leq B$$

$$a_1 z_1 + a_2 z_2 + \cdots a_M z_M \leq b$$

Variables to assign

$$\max_z c^T z \text{ subject to}$$

c: a matrix (or vector) of learned parameters

1. linear constraints on z
2. Each z_k is an integer

ILP is NP-complete ☹

But there are still well-designed solvers ➔ it's *useful*

Also Cool: Extend ILP to IQP Integer Quadratic Programming (with Linear constraints)

$$AW \leq B$$

$$a_1 w_1 + a_2 w_2 + \cdots a_M w_M \leq b$$

$$\max_z \underbrace{z c^T z}_{} \text{ subject to}$$

1. linear constraints on W
2. Each w_k is an integer

Quadratic term of our variables *z*

Depending on c, IQP can be easy or hard (NP-hard) ☹

But there are still well-designed solvers ➔ it's *useful*

So, how do we solve ILPs and IQPs?

- Fundamentally, they're a type of constraint (and search!) problem
- "Branch and bound" is a very common technique (Poole & Mackworth, Ch 3.8.1)
  - DFS, but keep the cost of the best solution found
  - Prune a path if its current cost + heuristic cost are worse than the cost of the best solution found
- Think of this as path pruning (from search) + domain splitting (from CSPs)

https://www.cvxpy.org/examples/basic/mixed_integer_quadratic_program.html

Mixed-integer quadratic program

A mixed-integer quadratic program (MIQP) is an optimization problem of the form

$$\begin{array}{ll} \text{minimize} & x^TQx + q^Tx + r \\ \text{subject to} & x \in \mathcal{C} \\ & x \in \mathbf{Z}^n, \end{array}$$

where $x \in \mathbf{Z}^n$ is the optimization variable ($\mathbf{Z}^n$ is the set of n-dimensional vectors with integer-valued components), $Q \in \mathbf{S}_+^n$ (the set of $n \times n$ symmetric positive semidefinite matrices), $q \in \mathbf{R}^n$, and $r \in \mathbf{R}$ are problem data, and $\mathcal{C}$ is some convex set.

An example of an MIQP is mixed-integer least squares, which has the form

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|_2^2 \\ \text{subject to} & x \in \mathbf{Z}^n, \end{array}$$

where $x \in \mathbf{Z}^n$ is the optimization variable, and $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^m$ are the problem data. A solution $x^\star$ of this problem will be a vector in $\mathbf{Z}^n$ that minimizes $\|Ax - b\|_2^2$.

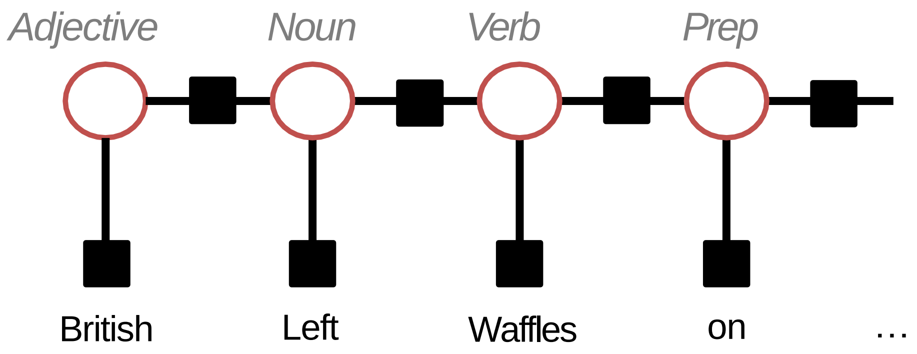
Example

In the following code, we solve a mixed-integer least-squares problem with CVXPY.

```
import cvxpy as cp
import numpy as np
```

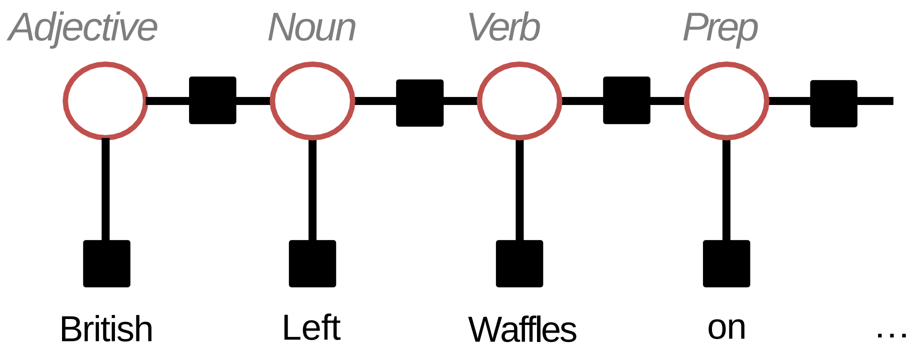
```
# Generate a random problem
np.random.seed(0)
m, n= 40, 25

A = np.random.rand(m, n)
b = np.random.randn(m)
```

Example: Sequence Tag Prediction as an IQP



Goal: Find the sequence of POS [part of speech] tags that maximize our score (given by previously learned weights)

Example: Sequence Tag Prediction as an IQP

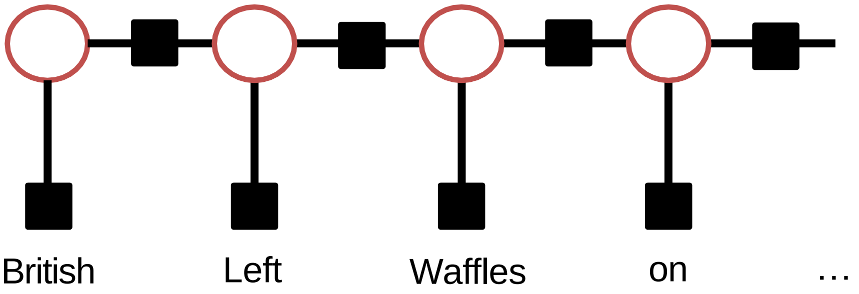


Goal: Find the sequence of POS [part of speech] tags that maximize our score (given by previously learned weights)

Put another way: We need to **find an assignment** of each word's POS that maximize the score, subject to sequence constraints

Example: Sequence Tag Prediction as an IQP

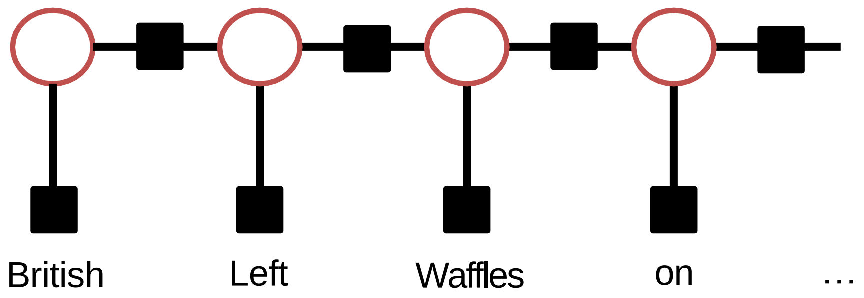
Put another way: We need to **find an assignment** of each word's POS that maximize the score, subject to sequence constraints



Core ideas:
(1) Our quadratic score function $f(w, z)$ will depend on previously learned weights w and the assignment to the structured z

Example: Sequence Tag Prediction as an IQP

Put another way: We need to **find an assignment** of each word's POS that maximize the score, subject to sequence constraints



Core ideas:

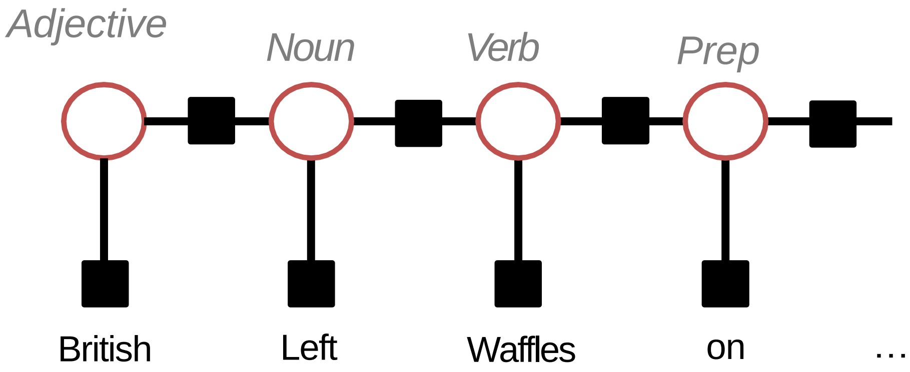
(1) Our quadratic score function $f(w, z)$ will depend on previously learned weights w and the assignment to the structured z

(2) The assignment to z must describe a **valid** sequence

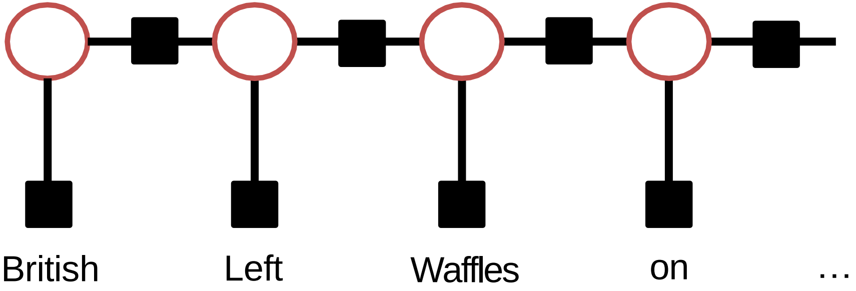
Example: Sequence Tag Prediction as an IQP

$\max_{z} f(w, z)$ subject to

1. linear constraints on z
2. Each z_* is an integer



The Big Representational Choice

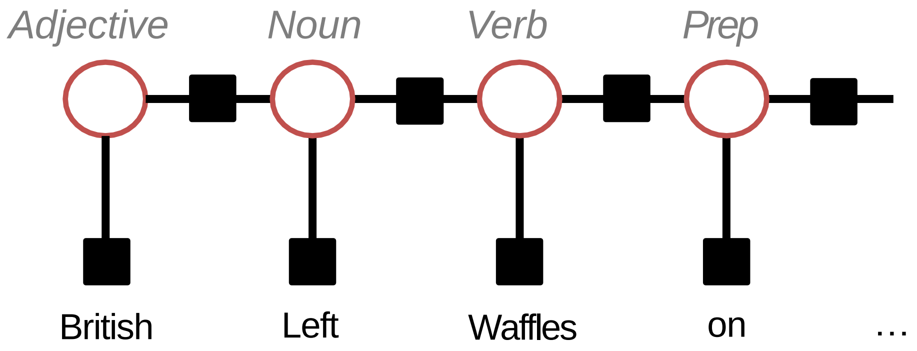


We are going to have a z for each possible pair of time step (word) i and POS tag k: $z_{i,k}$

Each $z_{i,k}$ will be binary: $z_{i,k} \in \{0,1\}$

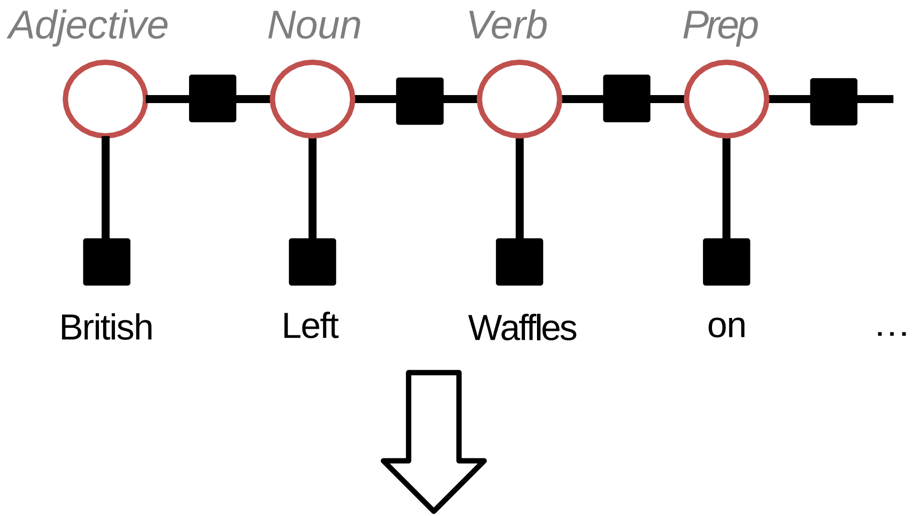
$z_{i,k}$ = 1 iff word i has tag k

Example of our Representation



$z_{1,\text{Noun}} = 0$	$z_{2,\text{Noun}} = 1$	$z_{3,\text{Noun}} = 0$	$z_{4,\text{Noun}} = 0$
$z_{1,\text{Verb}} = 0$	$z_{2,\text{Verb}} = 0$	$z_{3,\text{Verb}} = 1$	$z_{4,\text{Verb}} = 0$
$z_{1,\text{Adj.}} = 1$	$z_{2,\text{Adj.}} = 0$	$z_{3,\text{Adj.}} = 0$	$z_{4,\text{Adj.}} = 0$
…	…	…	…

Example of our Representation



Constraint 1: Each word must have at least one tag

$z_{1,\text{Noun}} = 0$
$z_{1,\text{Verb}} = 0$
$z_{1,\text{Adj.}} = 1$
…

$z_{2,\text{Noun}} = 1$
$z_{2,\text{Verb}} = 0$
$z_{2,\text{Adj.}} = 0$
…

$z_{3,\text{Noun}} = 0$
$z_{3,\text{Verb}} = 1$
$z_{3,\text{Adj.}} = 0$
…

$z_{4,\text{Noun}} = 0$
$z_{4,\text{Verb}} = 0$
$z_{4,\text{Adj.}} = 0$
…

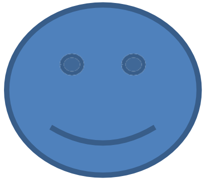
Constraint 2: Each word must have only one tag

Example: Sequence Tag Prediction as an IQP

$\max_{z} f(w, z)$ subject to

1. linear constraints on z
2. Each z_* is an {0, 1} integer

(One state per time) $\sum_k z_{ik} = 1$ for each time *i*



Example: Sequence Tag Prediction as an IQP

$\max_{z} f(w, z)$ subject to

1. linear constraints on z
2. Each z_* is an {0, 1} integer

$\sum_k z_{ik} = 1$ for each time i

score(Instance of data (“datum”))

scoring model

$$\text{score}_{\theta}(\ \text{Instance of dat}\ (\text{“datum”})\ )$$

$$\Downarrow$$

objective

$$F(\theta)$$

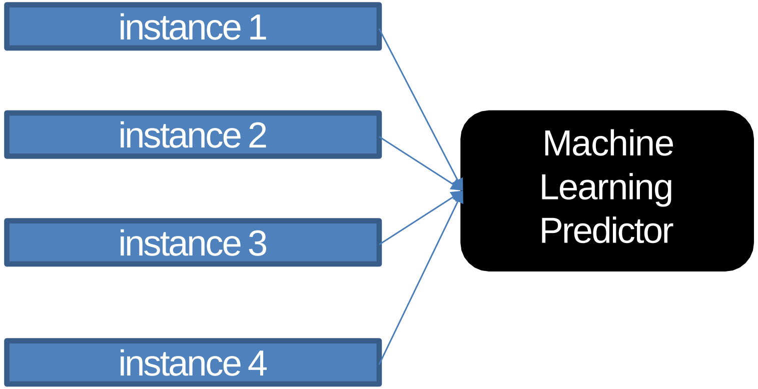
scoring model $\quad \text{score}_{\theta}($ Instance of data (“datum”) $)$

$\Downarrow$

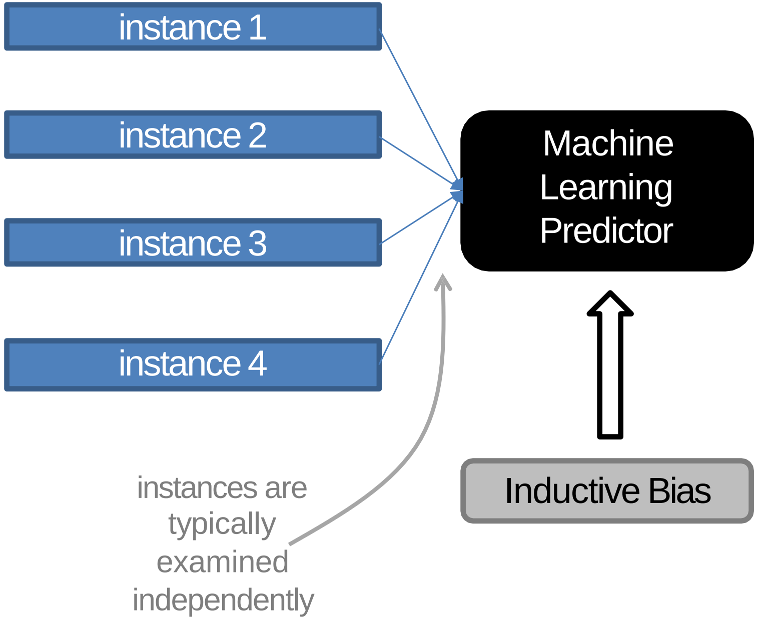
objective $\quad F(\theta)$

(implicitly) dependent on the observed data X=

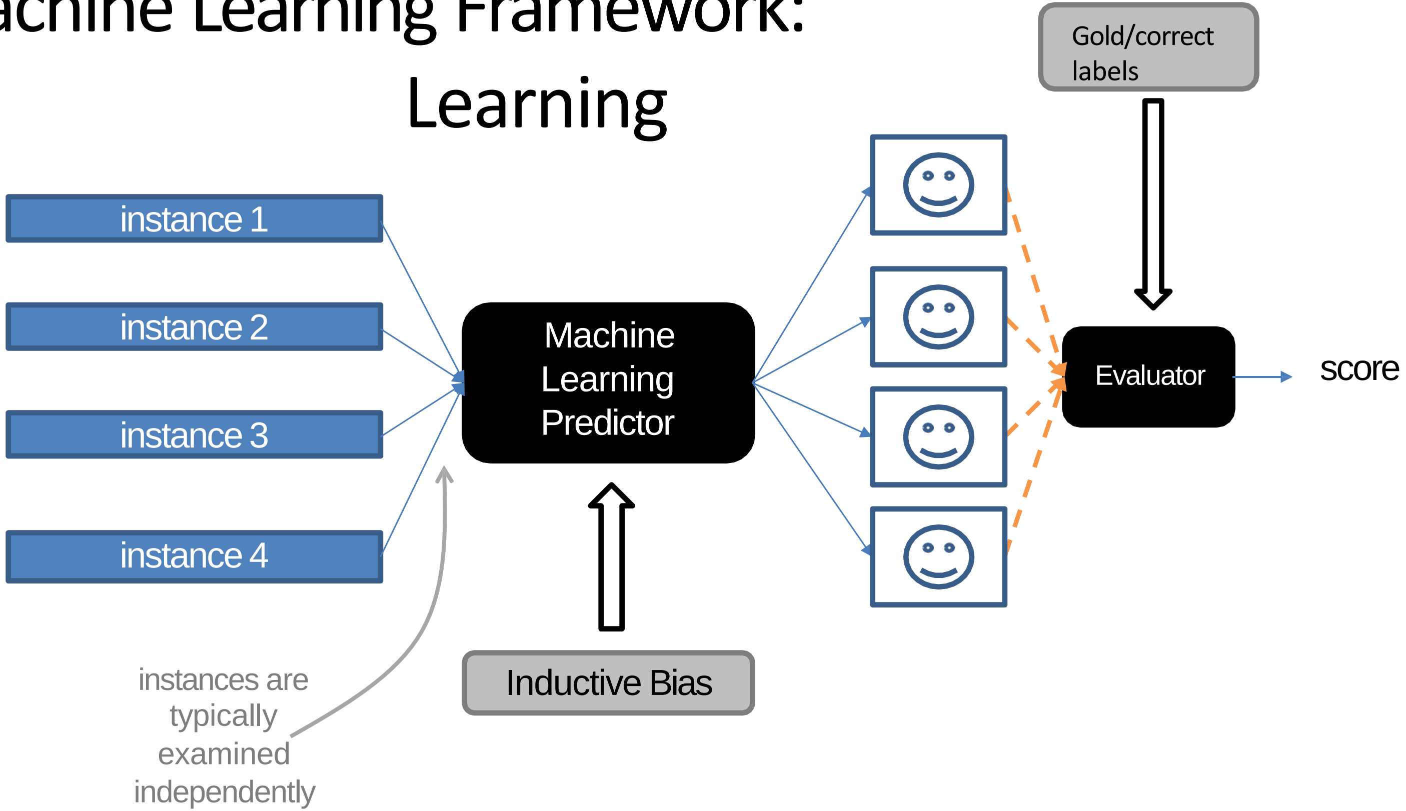
Machine Learning Framework: Learning

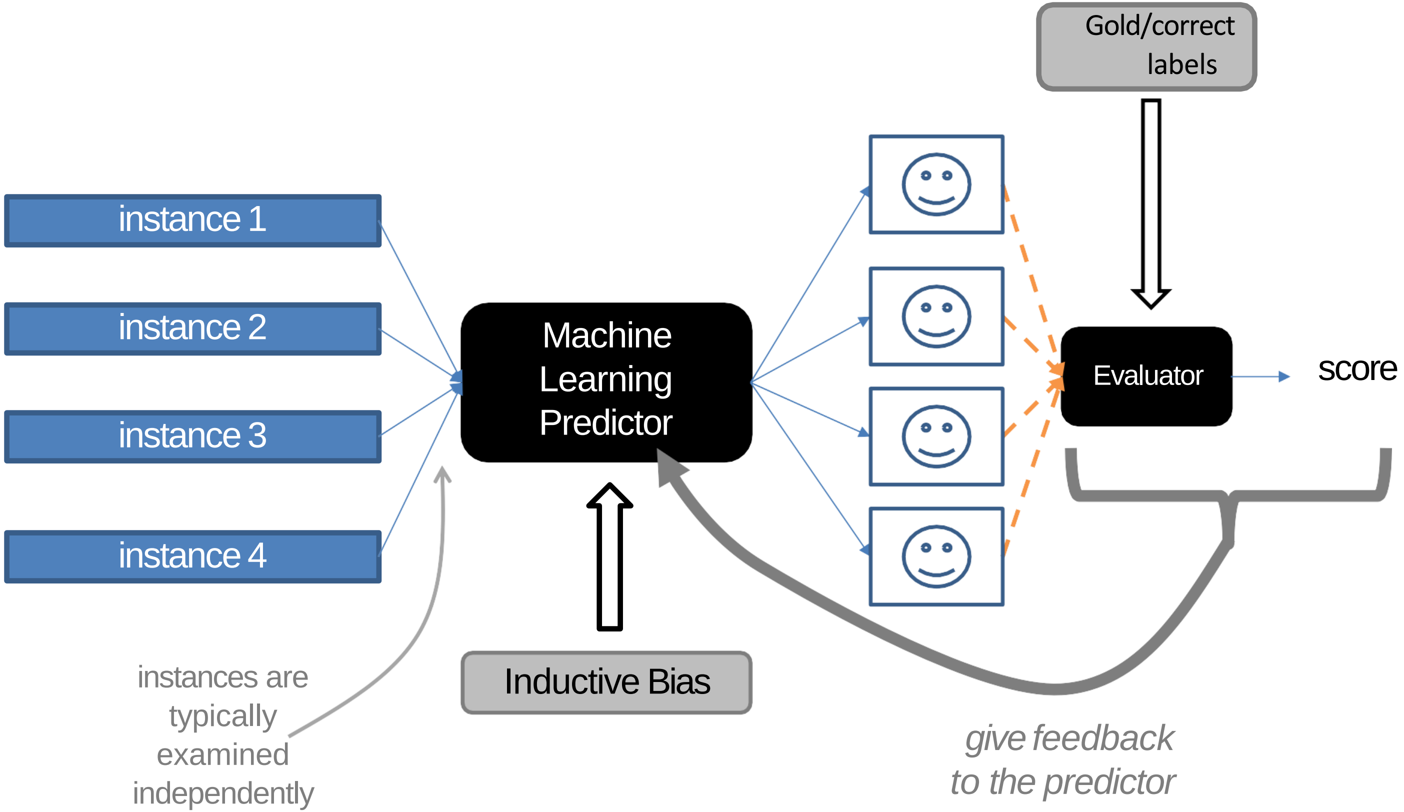


Machine Learning Framework: Learning



Machine Learning Framework: Learning




instance 1
instance 2
instance 3
instance 4
Machine Learning Predictor
Gold/correct labels
Evaluator
score
Inductive Bias
instances are typically examined independently
give feedback to the predictor

Classify with Goodness

$$\text{predicted label} = \underset{\text{label}}{\arg\max}\ \text{score}(\text{example}, \text{label})$$