Expanding the Horizons of NLP: Opportunities and Challenges

Vivek Srikumar

THE UNIVERSITY OF UTAH

This talk includes collaborative work with many people, including Zac Imel, Michael Tanana, Dan Roth, Tao Li and Nathan Schneider and the adposition gang.

The Wright Flyer

The pinnacle of aviation in 1905

We have traveled far in 100 years

We have traveled far in 100 years

Photo: Don-vip - Own work, CC BY-SA 4.0

Freud’s therapy room

The pinnacle of mental health in 1905

How far have we traveled in 100 years?

How far have we traveled in 100 years?

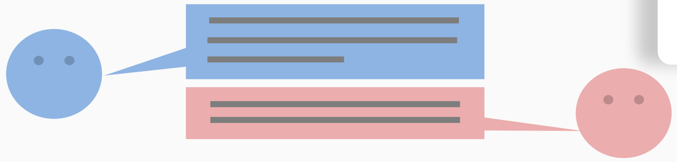


Photo: Jeff Hitchcock - Own work, CC BY 2.0

This talk

Natural language processing has the tremendous opportunity to revolutionize many fields,

…but we need to overcome several challenges.

This talk

Natural language processing has the tremendous opportunity to revolutionize many fields,

…but we need to overcome several challenges.

- Empathy and the Machine: A case study of NLP and Mental Health

This talk

Natural language processing has the tremendous opportunity to revolutionize many fields,

…but we need to overcome several challenges.

- Empathy and the Machine: A case study of NLP and Mental Health

- Challenge 1: The big [data problem]

This talk

Natural language processing has the tremendous opportunity to revolutionize many fields,

…but we need to overcome several challenges.

- Empathy and the Machine: A case study of NLP and Mental Health
- Challenge 1: The big [data problem]
- Challenge 2: Scaling NLP for everyone

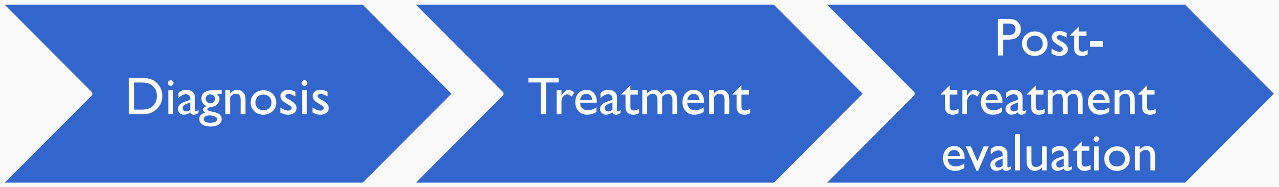
Michael Tanana

Zac Imel

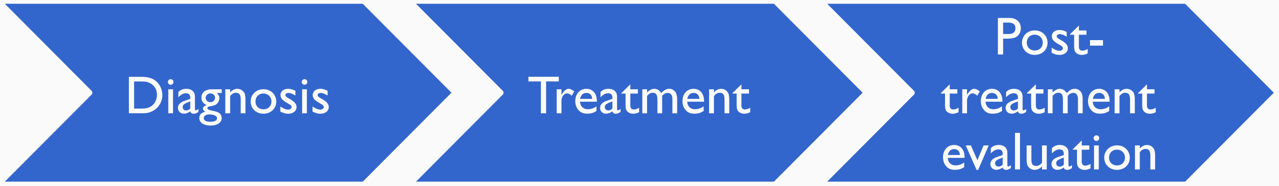
Empathy and the Machine

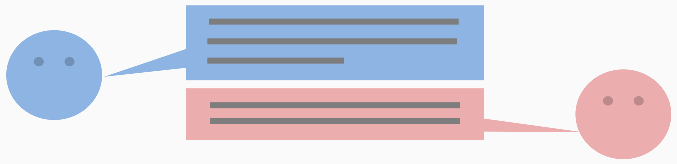
How can NLP help mental health?

A trip to a therapist…

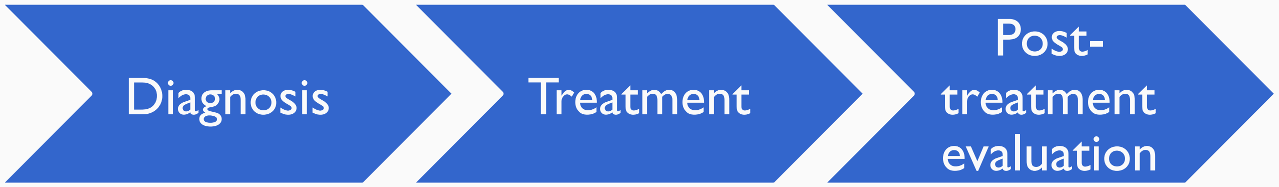


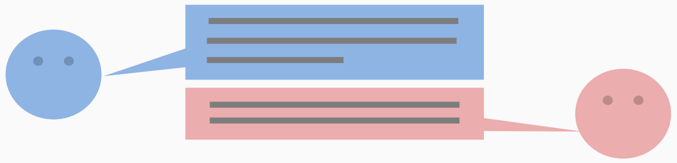
A trip to a therapist…



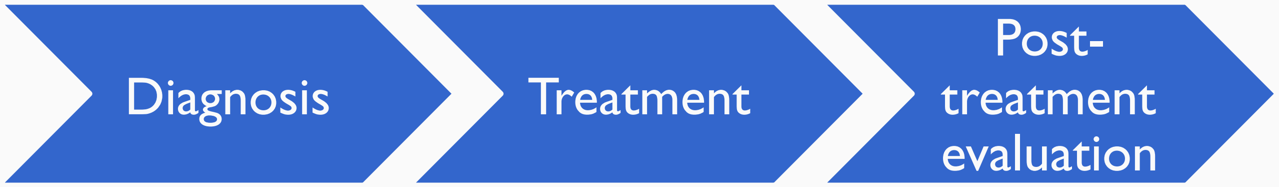


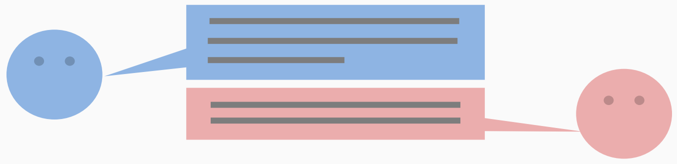
A trip to a therapist…





A trip to a therapist…





Mental health therapy

Essentially a dialogue between two people

- Are some treatments better than others?
- Are some counselors better than others?
- Can we measure patient improvements?
- How does the therapy process lead to changed behavior?

A collection of NLP problems

- Can we make sense of the hours of unstructured, often emotional dialog?
  - Automatically analyze, improve and study the effect of psychotherapy?

A collection of NLP problems

- Can we make sense of the hours of unstructured, often emotional dialog?
  - Automatically analyze, improve and study the effect of psychotherapy?

- **Several opportunities**:
  - Assessing therapy sessions [Tanana et al 2016]
  - Tracking sentiment [Tanana et al 2015]
  - Helping counselors *during* therapy
  - Helping train better therapists
  - Many more…

A therapy session transcript

Counselor: How do you feel about your progress so far?

Patient: Everyone's getting on me about my drinking

Counselor: Kind of like a bunch of crows pecking at you?

Patient: I'm not sure I can finish treatment.

Counselor: You're not sure you can finish treatment.

Patient: I drank a couple of times this week when I was with my brother.
I want to quit so badly,
but I don't think I can do it.

Example task:
Is a therapist being helpful?

Therapist's utterance can be a `simple reflection`, `complex reflection`, `open question`, `closed question`, etc.

Patient utterance can be about `sustaining` or `changing` their behavior or `neutral`.

These labels form the basis of a set called the MISC (Motivational Interviewing Skill Codes). [Houck et al 2012]

MISC coded session

[Houck et al 2012]

Counselor: How do you feel about your progress so far? `[Open Question]`

Patient: Everyone's getting on me about my drinking `[Follow-Neutral]`

Counselor: Kind of like a bunch of crows pecking at you? `[Complex Reflection]`

Patient: I'm not sure I can finish treatment. `[Sustain talk]`

Counselor: You're not sure you can finish treatment. `[Simple Reflection]`

Patient: I drank a couple of times this week when I was with
my brother. `[Sustain Talk]`
I want to quit so badly, `[Change Talk]`
but I don't think I can do it. `[Sustain Talk]`

How therapists get feedback today

1. Patient and doctor talk to each other
2. Transcribe the conversation
3. Label every utterance of the transcript as being in line with the counseling style
4. Generate aggregate statistics from the labels
5. Give feedback to the therapist

How therapists get feedback today

1. Patient and doctor talk to each other
2. Transcribe the conversation
3. Label every utterance of the transcript as being in line with the counseling style
4. Generate aggregate statistics from the labels
5. Give feedback to the therapist

Does not scale

Can we automatically label utterances?

[Tanana et al 2016, JSAT]

The hope

To provide automatic feedback to therapists after every session

To train better therapists

To conduct aggregate studies about the state of mental health treatment

The therapy dataset

- 341 therapy sessions
  - approximately 1.7 million words,
  - 175,000 utterances

- Focus on substance abuse
  - Affects 21 million Americans (as of 2014)

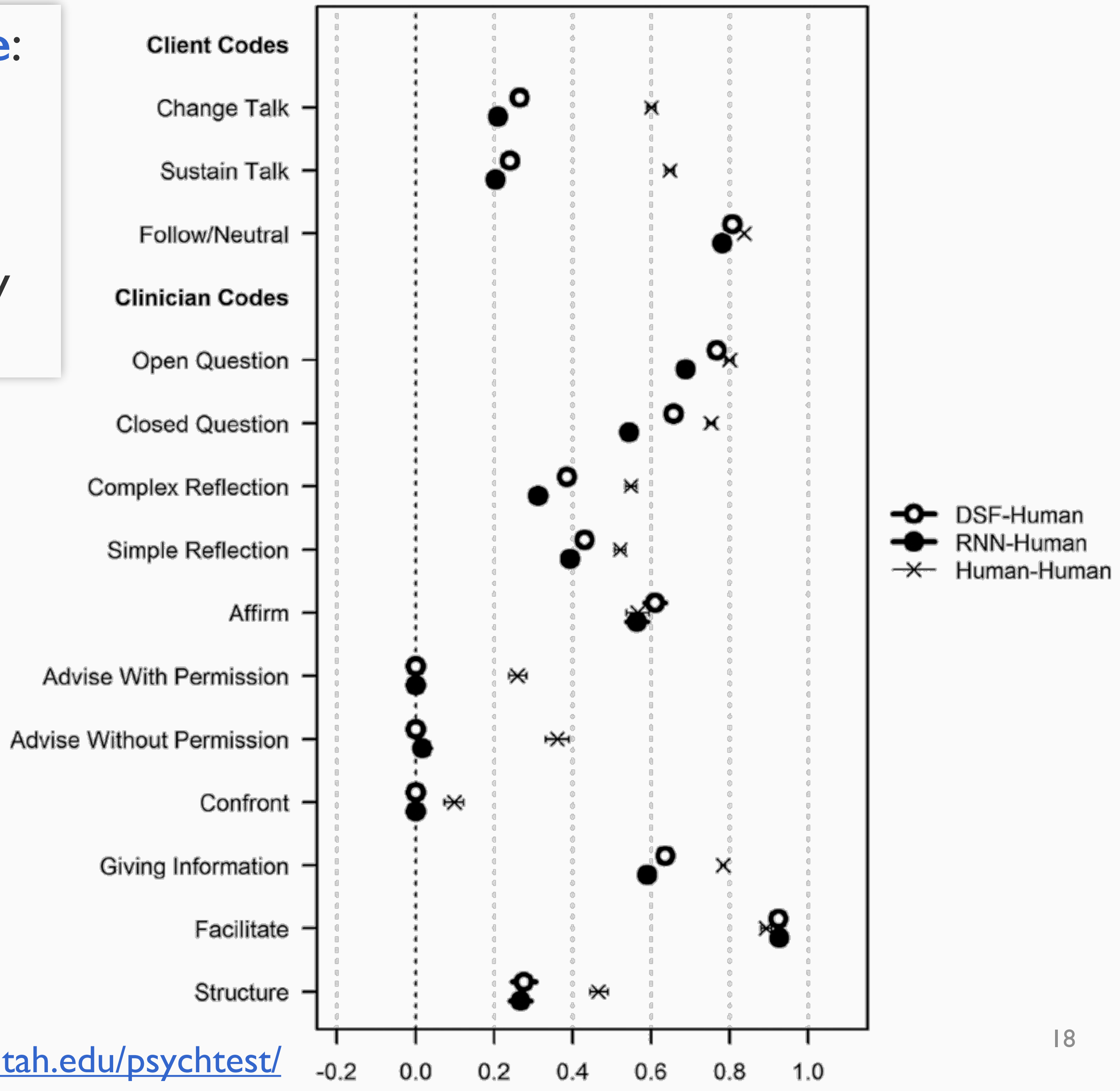
We trained two sequence models

- Both models label utterances in the session with one of the MISC labels
  - Differ in their feature representation of each utterance

- Model 1:
  - A traditional feature based model
  - Based on words and dependency features

- Model 2:
  - A recursive neural network for each sentence
  - Operates on top of the dependency tree

The punchline: Trained models predicted many labels similar to human reliability on the test set.

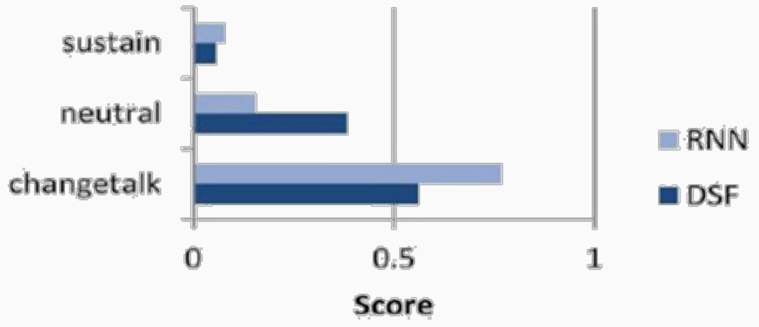


Live demo: http://sri.utah.edu/psychtest/

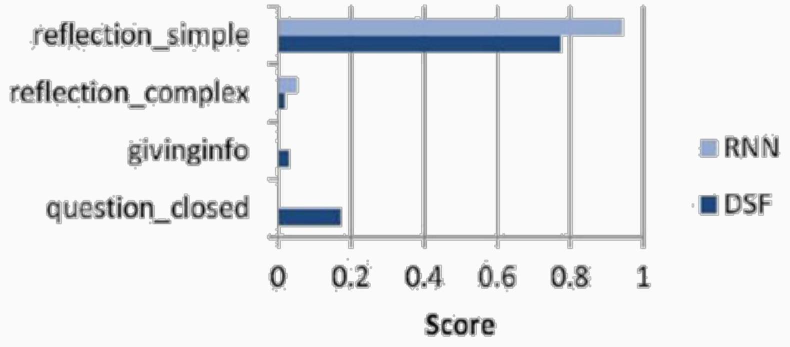
Client

Clinician

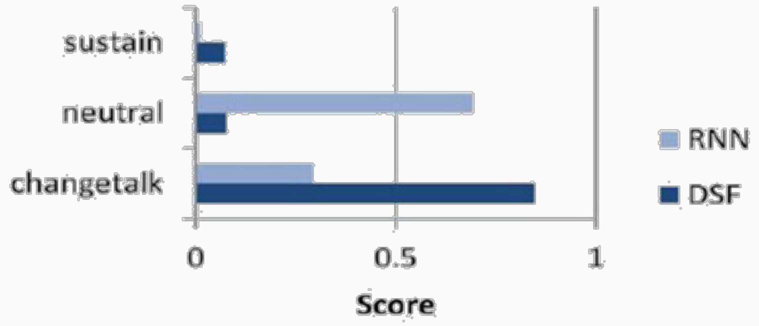
(a) "and just sometime i just get tired of being high so" (change talk)



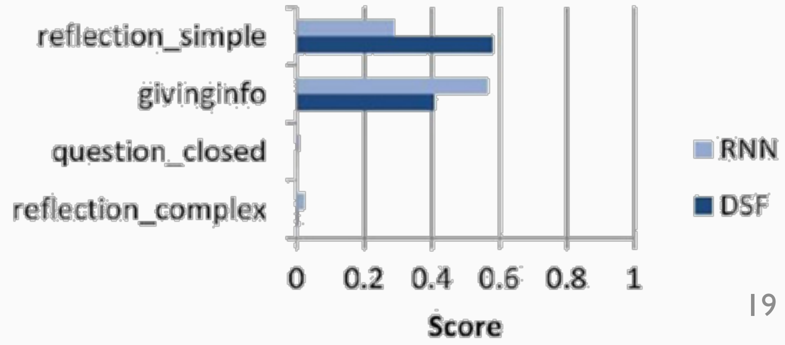
(c) "you said that you graduated and that you did accomplish some of your goals it sounds like" (reflection: simple)



(b) "but something happened in that motel where i just said i just can't do this anymore" (change talk)



(d) "it seems like you were drinking seven days a week about four drinks at a time" (giving info)



Natural Language Processing can transform many fields. But…

Still a long way to go

Challenges abound

Wanted

Programs that can learn to understand and reason about the world via language

Challenges

- Representations
  - *What is a* ***good*** representation of language?
    - Eg: How can we best represent words/sentences/utterances to reason about complex text like the therapy transcripts?
- Data
  - How do we get around the need for annotated data?
    - Annotated data is a precious resource
- Efficiency
  - How can we scale predictive models (eg: to every doctor and patient in the world)?
    - How can we avoid unnecessary computation?

Challenges

- Representations

Linguistic intuitions can help develop useful *structured* representations for performing reasoning about text

- Data
  - How do we get around the need for annotated data?
    - Annotated data is a precious resource

This talk

- Efficiency
  - How can we scale predictive models (eg: to every doctor and patient in the world)?
    - How can we avoid unnecessary computation?

Whence Data?

Most successes of data-driven computing made possible by supervised methods.

Data is a precious commodity.

Two kinds of data scarcity problems

- 1. For medical applications, even unlabeled data is difficult to obtain
    - *Public data → more accountable science*
    - How can we create publicly accessible datasets for health care applications and also preserve patient privacy?
    - *An important open question*

  2. Annotated data is a precious commodity

Annotated data is precious

- Standard practice in NLP, computer vision, etc
  - Annotate datasets for tasks that we care about
- *Is this sustainable?*
- A well crafted data set can take years to create
- Need better algorithmic solutions

Tao Li

Can we exploit easier to obtain signals?

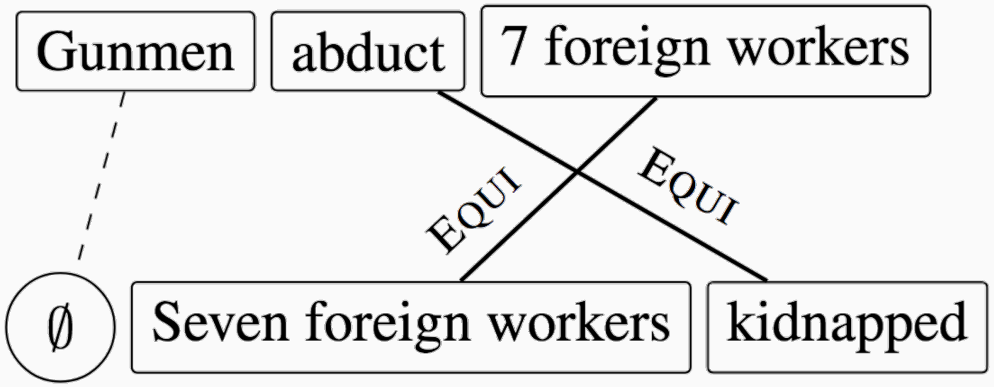
Many problems where we want to predict a structure, but

- We don’t have (much) data for that representation
- We have data for a related phenomenon

Example 1: Aligning text

[Li & Srikumar 2016]

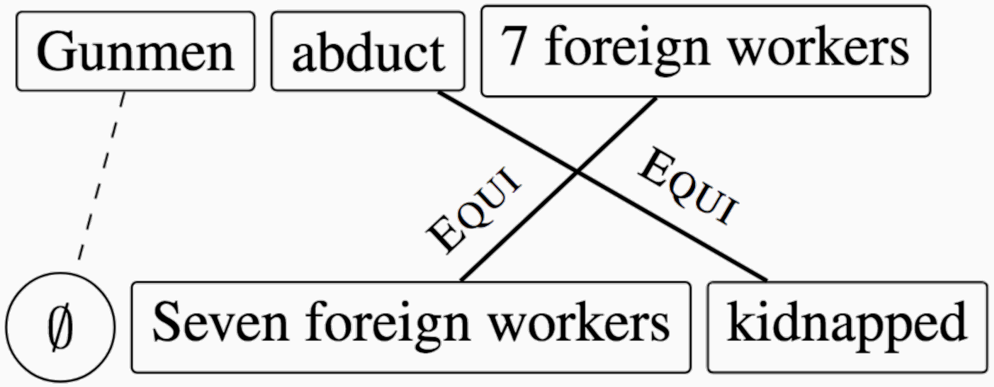
- Suppose we want to align phrases in text



Example 1: Aligning text

[Li & Srikumar 2016]

- Suppose we want to align phrases in text



- Alignments are tedious to annotate

- But several related tasks are cheaper to annotate

How similar are these two sentences?

[Agirre et al. 2012-2016]

NYPD Twitter Outreach Backfires Badly

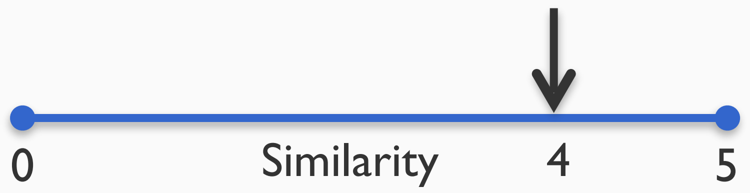
NYPD's social media campaign a flop?

How similar are these two sentences?

[Agirre et al. 2012-2016]

NYPD Twitter Outreach Backfires Badly

NYPD's social media campaign a flop?

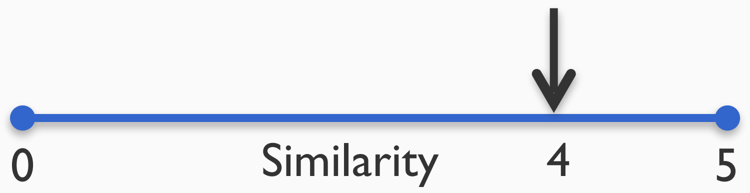


How similar are these two sentences?

[Agirre et al. 2012-2016]

NYPD Twitter Outreach Backfires Badly

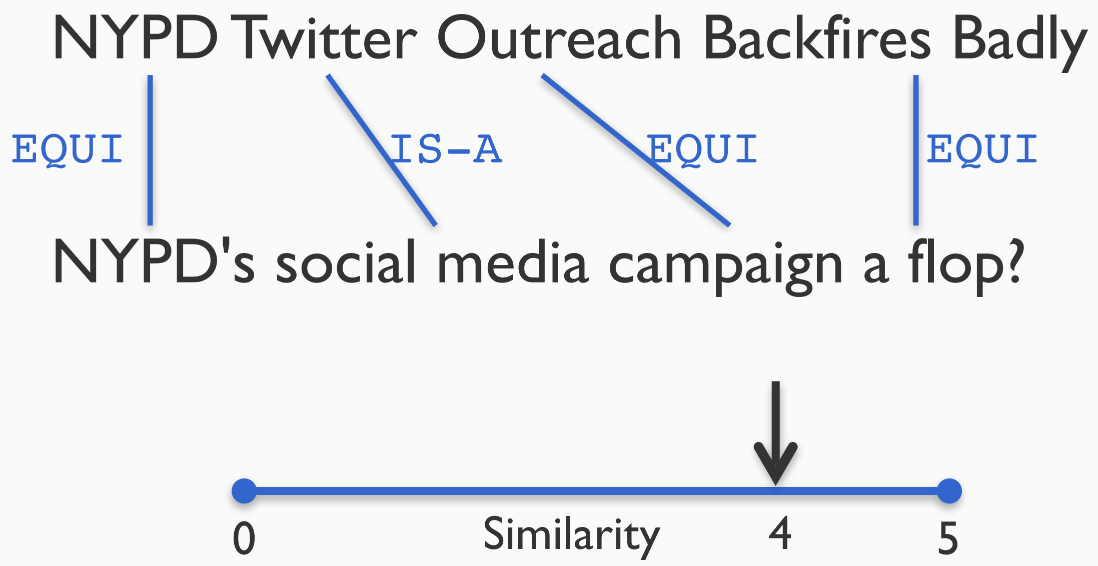
NYPD's social media campaign a flop?



But why? We can align their constituents to get a certificate.

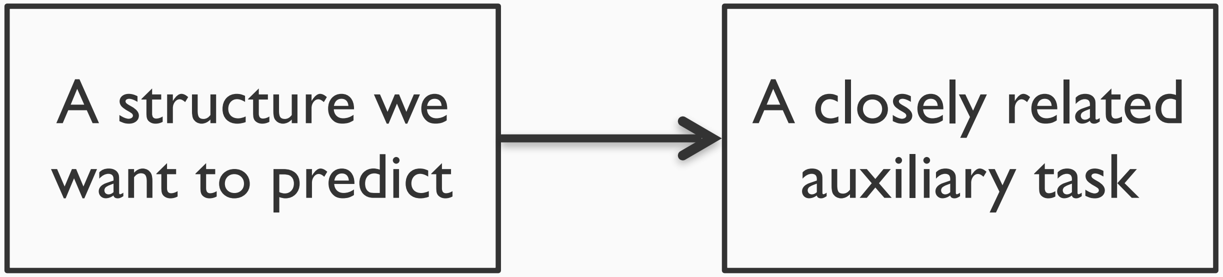
How similar are these two sentences?

[Agirre et al. 2012-2016]


NYPD Twitter Outreach Backfires Badly
EQUI
IS-A
EQUI
EQUI
NYPD's social media campaign a flop?
0
Similarity
4
5

But why? We can align their constituents to get a certificate.

What we have...


A structure we want to predict
A closely related auxiliary task

Not enough training data

Alignments

Lot of training data

Sentence similarities

The structure is a certificate for the auxiliary prediction

Example 2:
Prepositions trigger semantic relations

They ran **to** the roof **for** a quick escape.

```
Destination   Purpose
```

They ran **for** the roof **to** escape the cops.

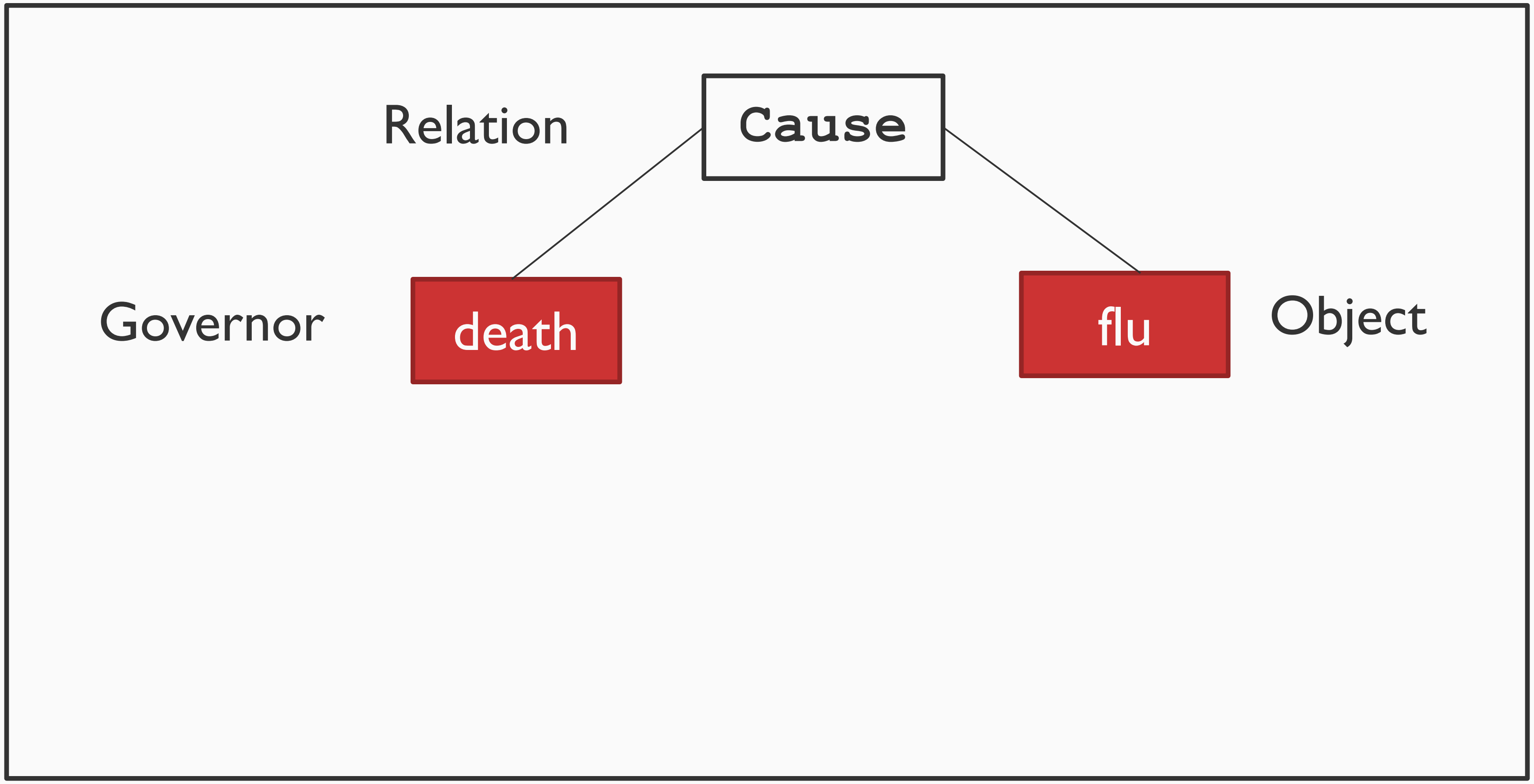
[Srikumar & Roth 2013],
[Schneider et al 2015, 2016],
[Hwang et al 2017]

Slide credit: Nathan Schneider

Structure of preposition relations

[Srikumar & Roth 2013]

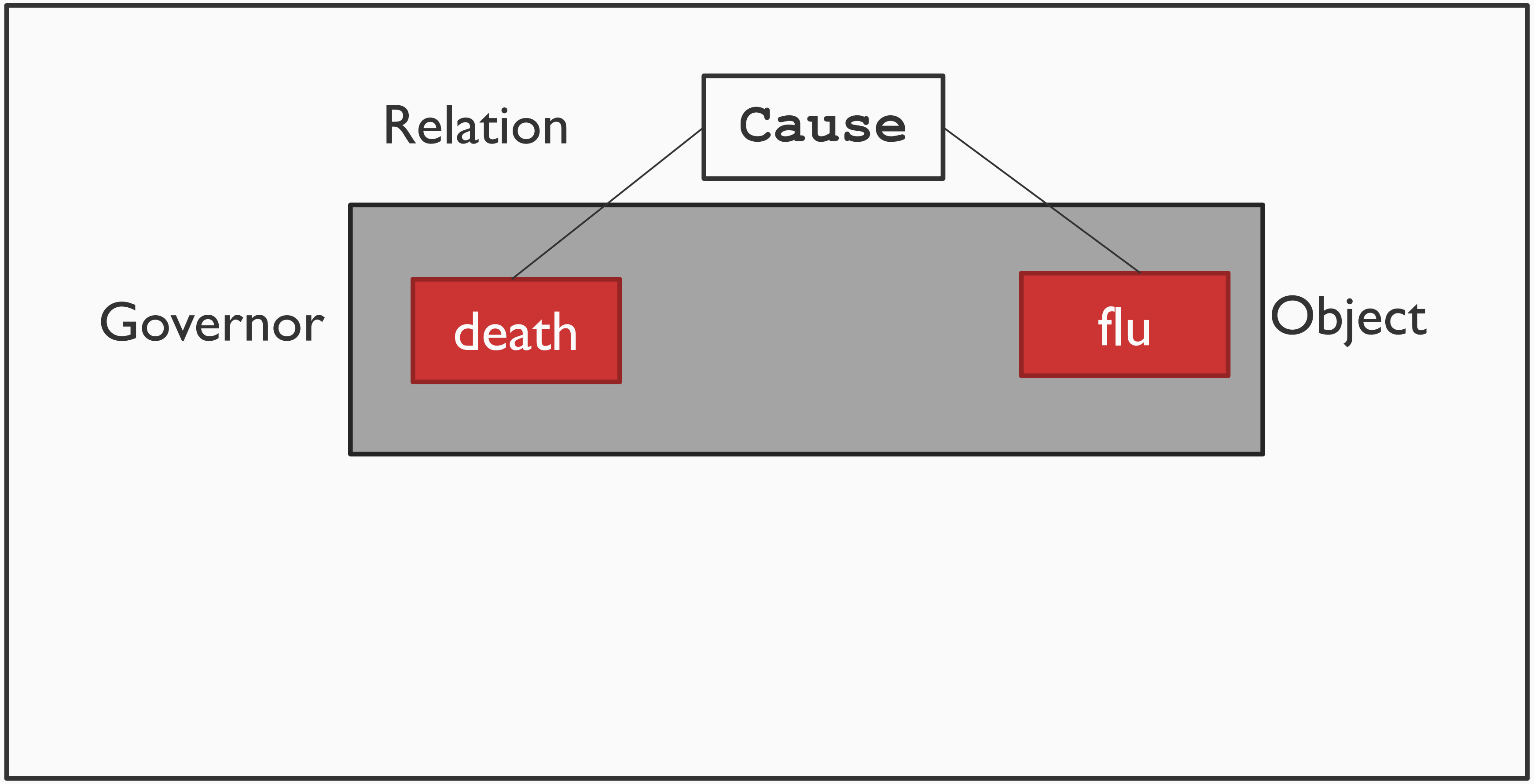
Poor care led to her death **from** flu.



Structure of preposition relations

[Srikumar & Roth 2013]

Poor care led to her death **from** flu.



Relation depends on argument types

Poor care led to her death **from** flu.

`Cause`(death, flu)

Relation depends on argument types

Poor care led to her death **from** flu.

`Cause`(death, flu)

Poor care led to her death **from** pneumonia.

How do we generalize the classifier to unseen arguments in the same "type"?

Abstract *flu and pneumonia* into the same group

WordNet `IS-A` hierarchy

```
pneumonia
  => respiratory disease
    => disease
      => illness
        => ill health
          => pathological state
            => physical condition
              => condition
                => state
                  => attribute
                    => abstraction
                      => entity
```

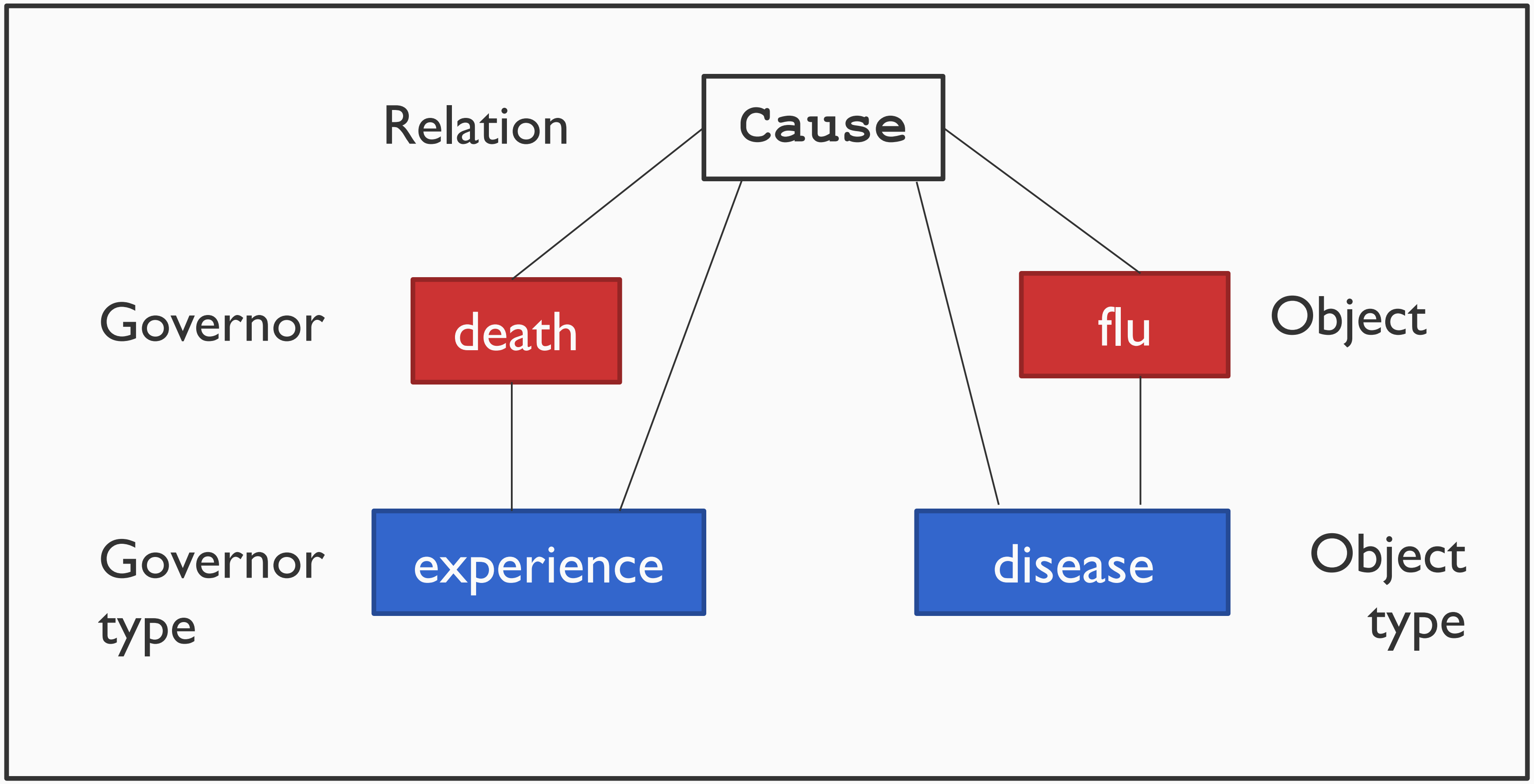
Picking the right level in this hierarchy can generalize pneumonia and flu

More general, but less discrimniative

Picking incorrectly will over-generalize

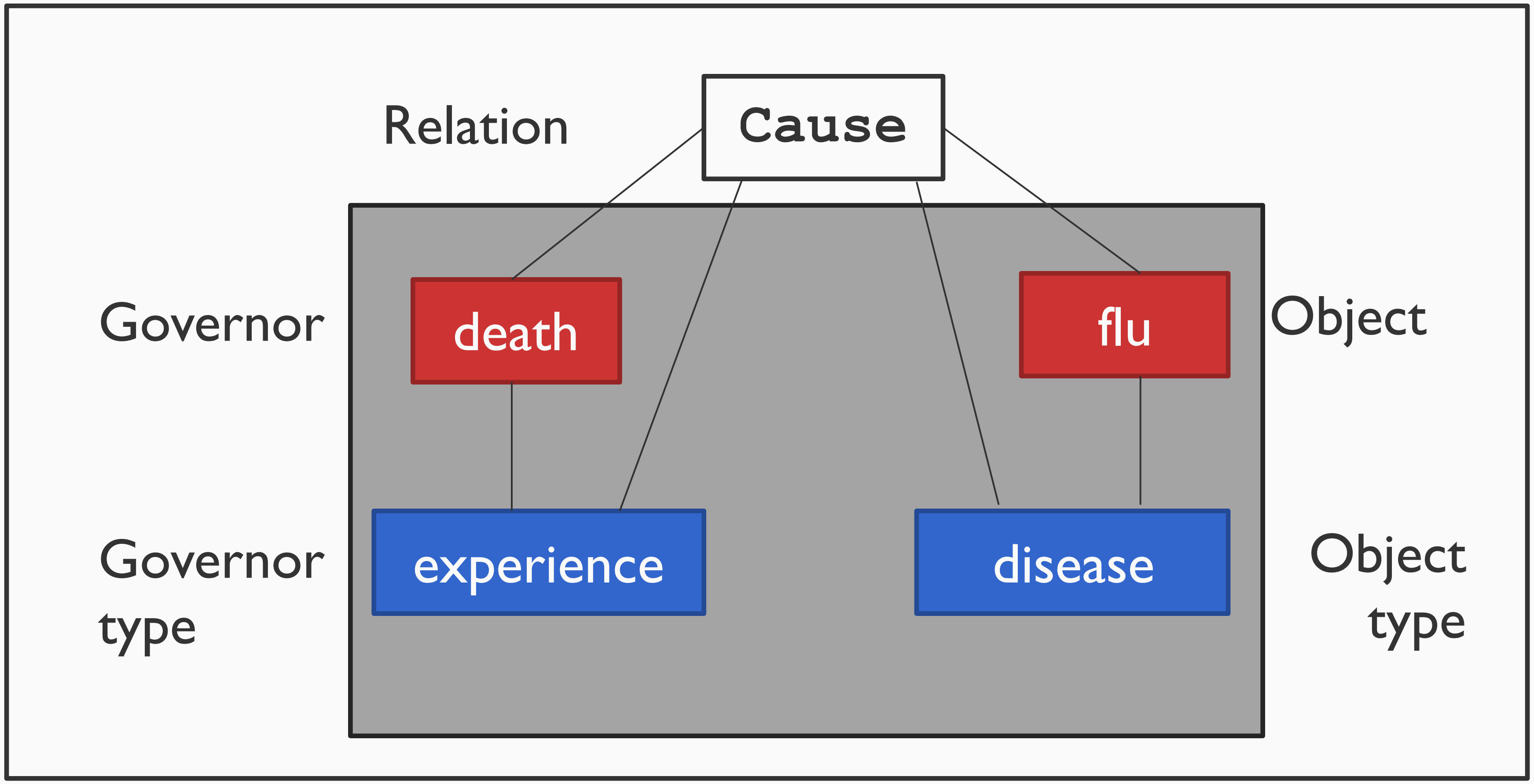
Structure of prepositions

Poor care led to her death **from** flu.



Structure of prepositions

Poor care led to her death **from** flu.



What we have...

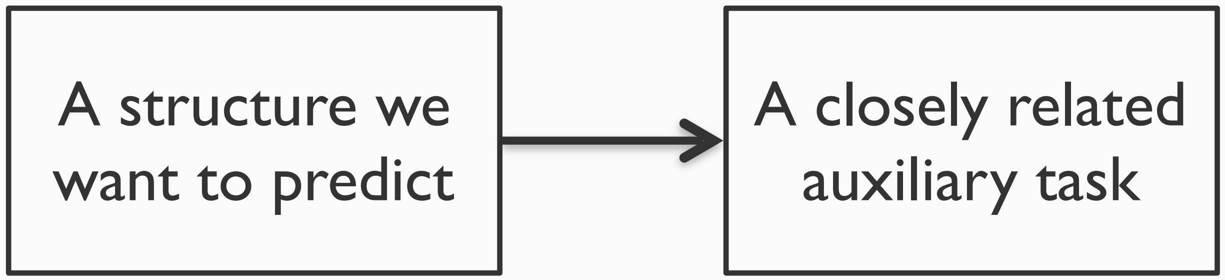
What we have...

A structure we want to predict

No training data

Preposition relation, arguments, types

What we have...


A structure we want to predict
A closely related auxiliary task

No training data

Preposition relation, arguments, types

Training data available

Preposition relation alone

The structure is a certificate for the auxiliary prediction

Joint prediction to the rescue

- Predict both the structure and the auxiliary output jointly
  - Predict both alignment and sentence similarity in one shot
  - Global inference guided by constraints

- Seen this way, *we don't have two tasks*
  - Both tasks are part of a larger whole

Why joint prediction?

Suppose we have a joint scorer for alignments and sentence similarities

- For a sentence that is only labeled with sentence similarity, we can ask:
  - What is the best alignment that gets this similarity?

- Essentially, we can "complete" examples

Learning from Partially Labeled Data

Instantiated for the alignment case

1. Train only with alignment-only dataset

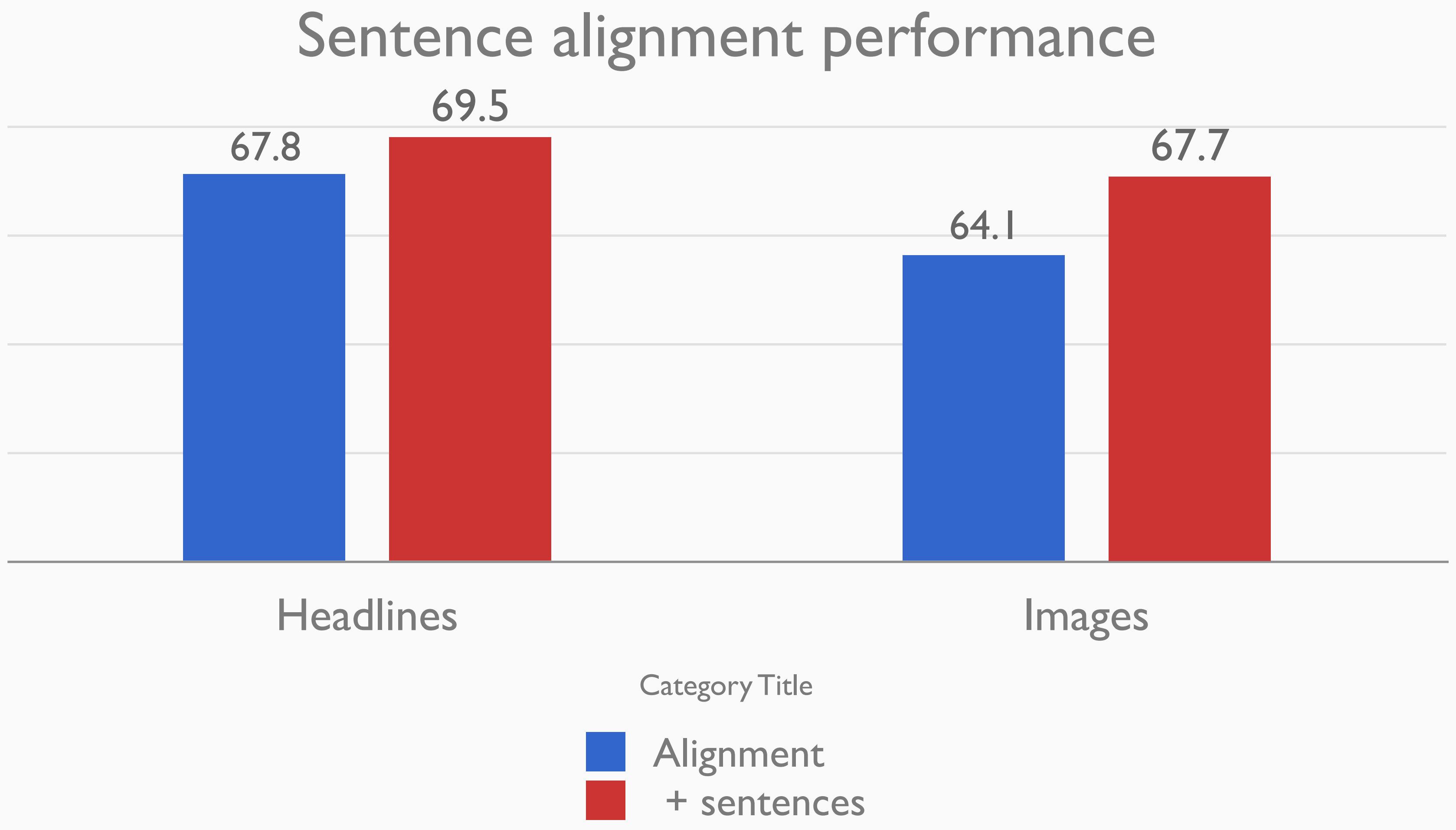
2. Predict sentence similarity for alignment-only data

3. Predict alignment for sentence-only data

4. Train on both:
   - Alignment examples (+ predicted sentence similarities)
   - Sentence similarity (+ predicted alignments)

Joint inference between the two tasks allows such learning.

How well does this work?



Key lessons

- Incidental supervision signals can help train structured classifiers
  - Need to identify relationships between structured classification task at hand and supervision signal
  - Tested for paraphrase detection, textual entailment, transliteration, information extraction, semantic textual similarities, preposition relation prediction [Chang et al 2010, Srikumar and Roth 2013, Li and Srikumar 2017]
- Constraints help modeling structured predictors
  - Internal self-consistency of prediction helps guide learning
  - Constraints can also act as weak supervision

Challenges

- Representations

  Linguistic intuitions can help develop useful *structured* representations for performing reasoning about text

- Data
  - Annotated data is a precious resource
  - How do we get around the need for annotated data?

- Efficiency
  - Predictive models need to scale (eg: to every doctor and patient in the world)
  - How can we avoid unnecessary computation?

Challenges

- Representations

  Linguistic intuitions can help develop useful *structured* representations for performing reasoning about text

- Data

  Exploit the need for consistency between different kinds of linguistic predictions to act as a surrogate for training data.

- Efficiency
  - Predictive models need to scale (eg: to every doctor and patient in the world)
  - How can we avoid unnecessary computation?

Scaling Human Language Technology

The challenge

- Natural language understanding needs to scale to every person, phone, document, utterance out there

- Example: For clinical applications, can we deploy them in every clinic or hospital in the world?

A ground up rethinking

- Faster inference [Srikumar et al 2012, Kundu et al 2013]
  - No need to solve certain inference problems if they are similar to ones we have already solved

- Faster feature extraction [Srikumar 2017]
  - Automatically refactor feature extraction to reduce redundant computation

- Faster dot products [Shafiee et al 2016]
  - Better machine level support, novel hardware

A ground up rethinking

- Faster inference [Srikumar et al 2012, Kundu et al 2013]
  - No need to solve certain inference problems if they are similar to ones we have already solved
- Faster feature extraction [Srikumar 2017]
  - Automatically refactor feature extraction to reduce redundant computation
- Faster dot products [Shafiee et al 2016]
  - Better machine level support, novel hardware

Faster feature extraction

Feature extraction has always been “somebody else’s problem”.

Can be time consuming and adds up over a classifier’s lifetime.

Faster feature extraction

Feature extraction has always been “somebody else’s problem”.

Can be time consuming and adds up over a classifier’s lifetime.

Can we make feature extraction faster?

Let us examine feature extraction

Example: Classifying words

The cat in a hat sat on the mat.

Example: Classifying words

The cat in a hat sat on the mat.

Some "typical" NLP features may include:

Example: Classifying words

The cat in a hat sat on the mat.

Some "typical" NLP features may include:

`word`: The surface form of a word $\xrightarrow{\text{produces}}$ {word=hat}

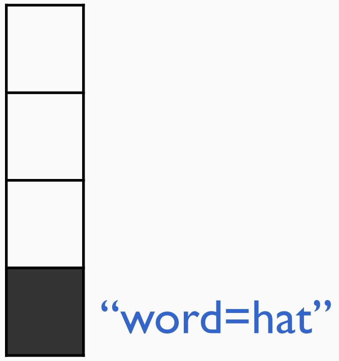
Example: Classifying words

The cat in a hat sat on the mat.

Some "typical" NLP features may include:

`word`: The surface form of a word $\xrightarrow{\text{produces}}$ {word=hat}

Really shorthand for the sparse vector that is zero everywhere except for one element whose basis corresponds to the string "word=hat"



Example: Classifying words

The cat in a hat sat on the mat.

Some "typical" NLP features may include:

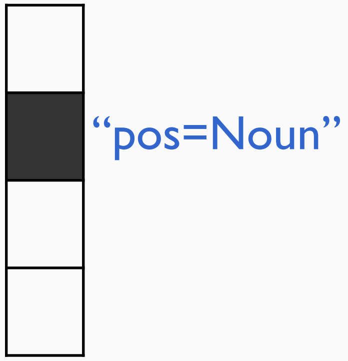
Example: Classifying words

The cat in a hat sat on the mat.

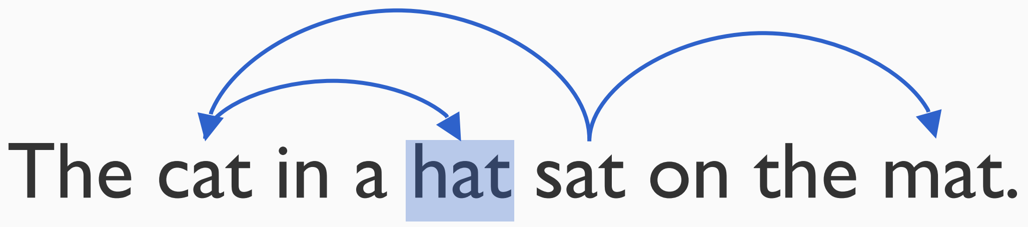
Some "typical" NLP features may include:

`pos`: Part of speech of a word $\xrightarrow{\text{produces}}$ {pos=Noun}

Really shorthand for the sparse vector that is zero everywhere except for one element whose basis corresponds to the string "pos=Noun"



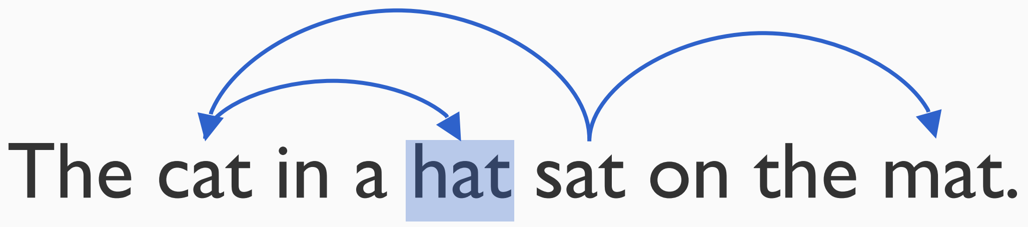
Example: Classifying words



The cat in a hat sat on the mat.

Some “typical” NLP features may include:

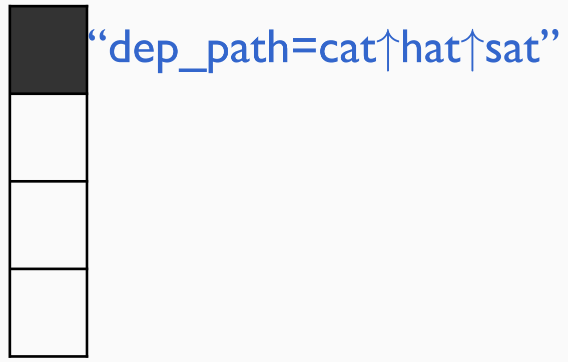
Example: Classifying words



Some "typical" NLP features may include:

`dep_path`: Dependency path to root $\xrightarrow{\text{produces}}$ {dep_path=cat↑hat↑sat}

Really shorthand for the sparse vector that is zero everywhere except for one element whose basis corresponds to the string "dep_path=cat↑hat↑sat"



Feature extractors are functions

`word`: The surface form of a word $\longrightarrow$ {word=hat}

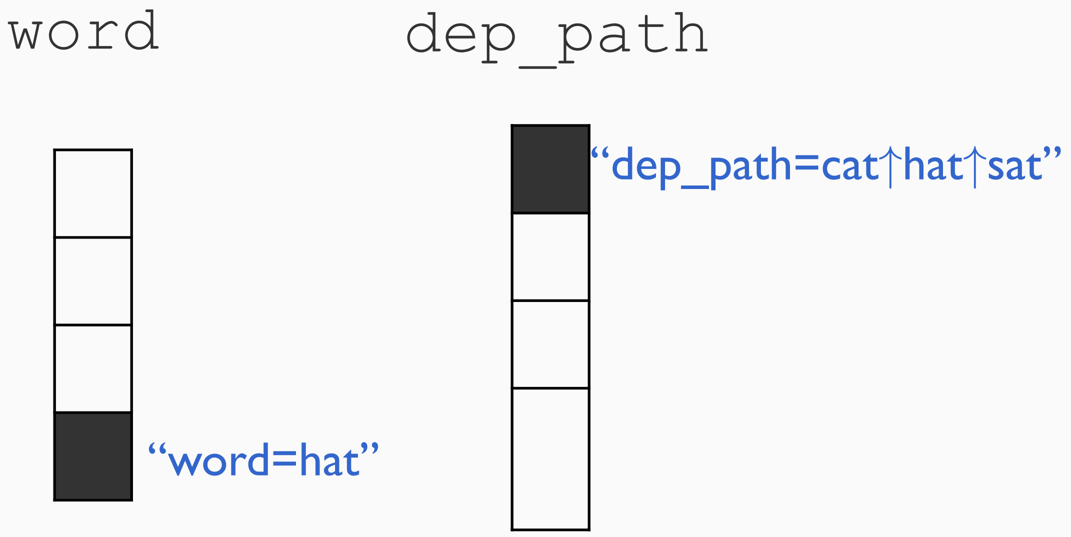
`pos`: Part of speech of a word $\longrightarrow$ {pos=Noun}

`dep_path`: Dependency path to root $\longrightarrow$ {dep_path=cat↑hat↑sat}

Functions from any objects (such as words, images, etc.) to a (possibly infinite dimensional) vector space

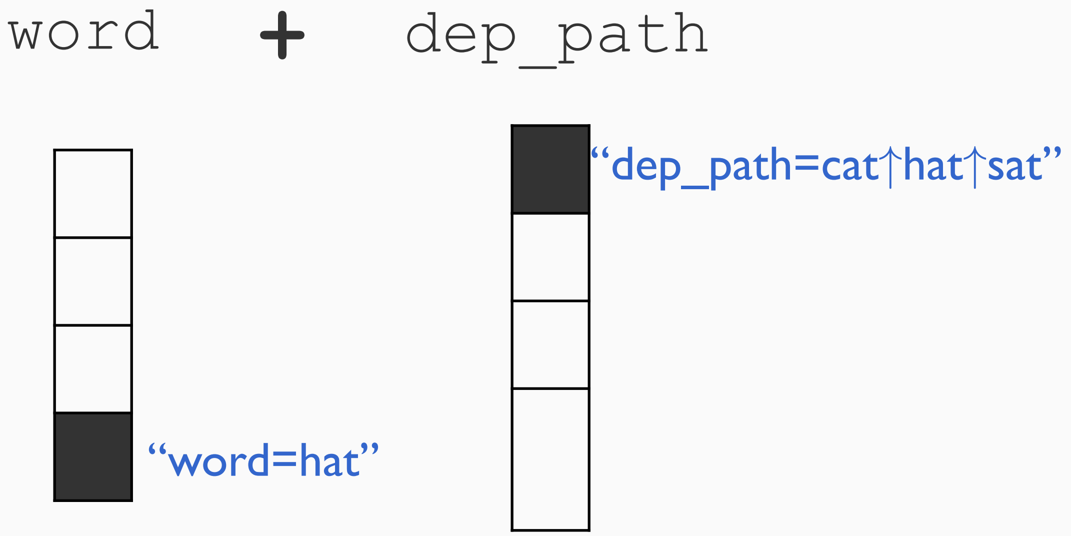
Operations on these functions

Feature addition



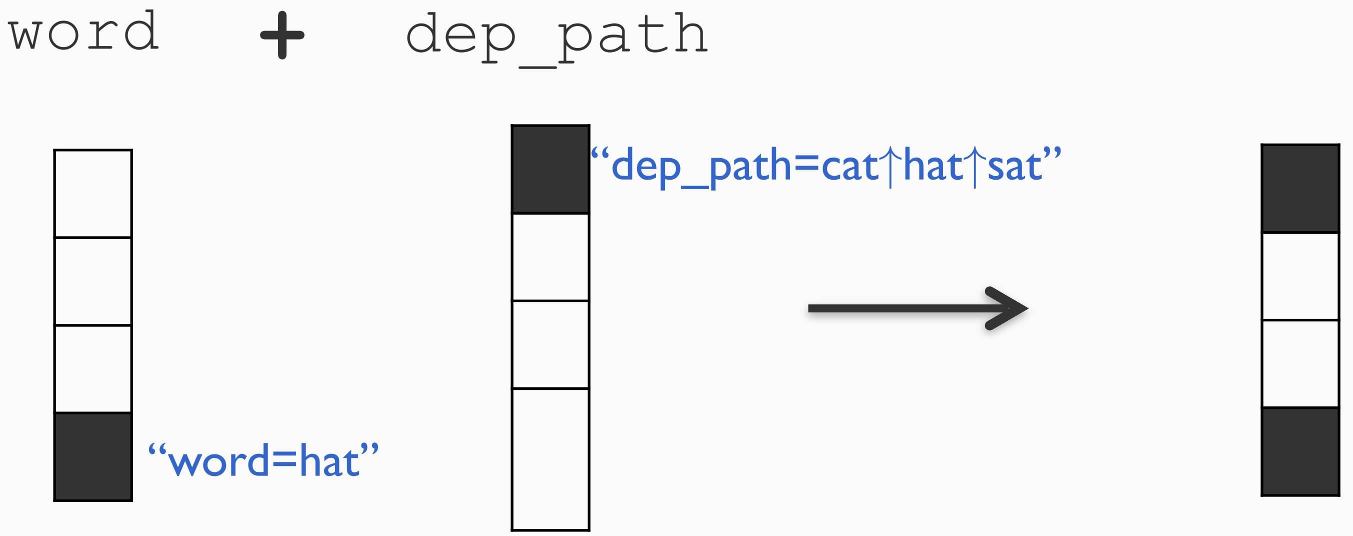
Operations on these functions

Feature addition



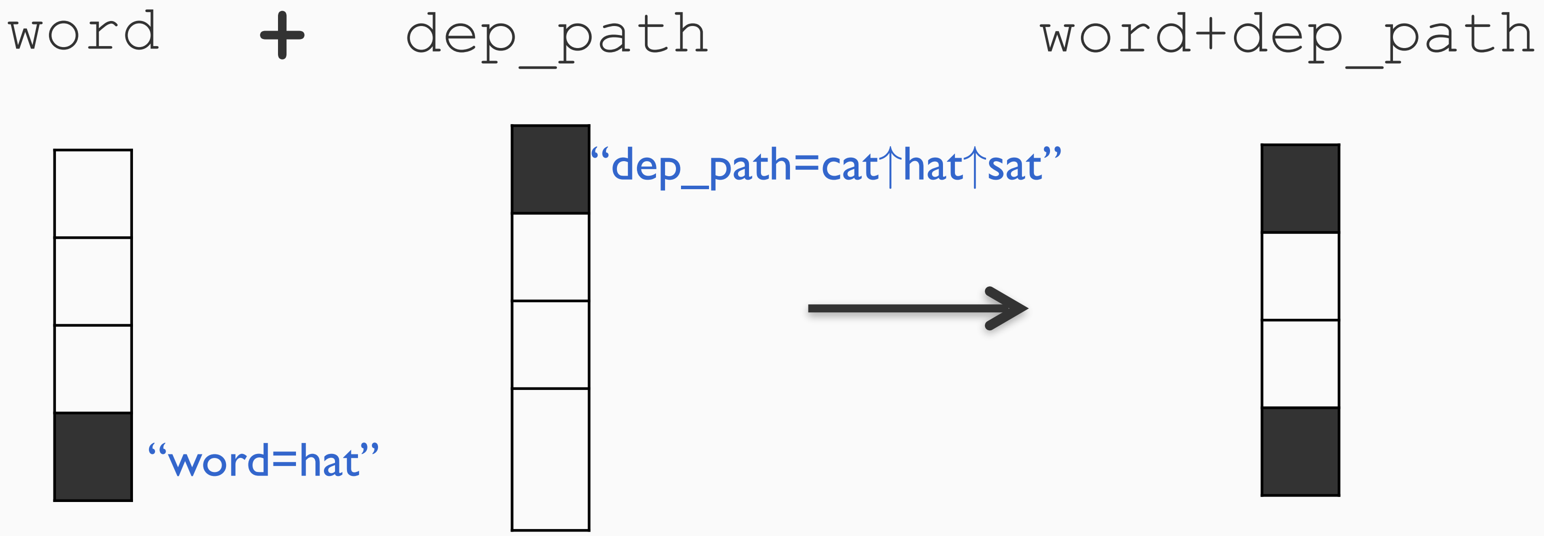
Operations on these functions

Feature addition



Operations on these functions

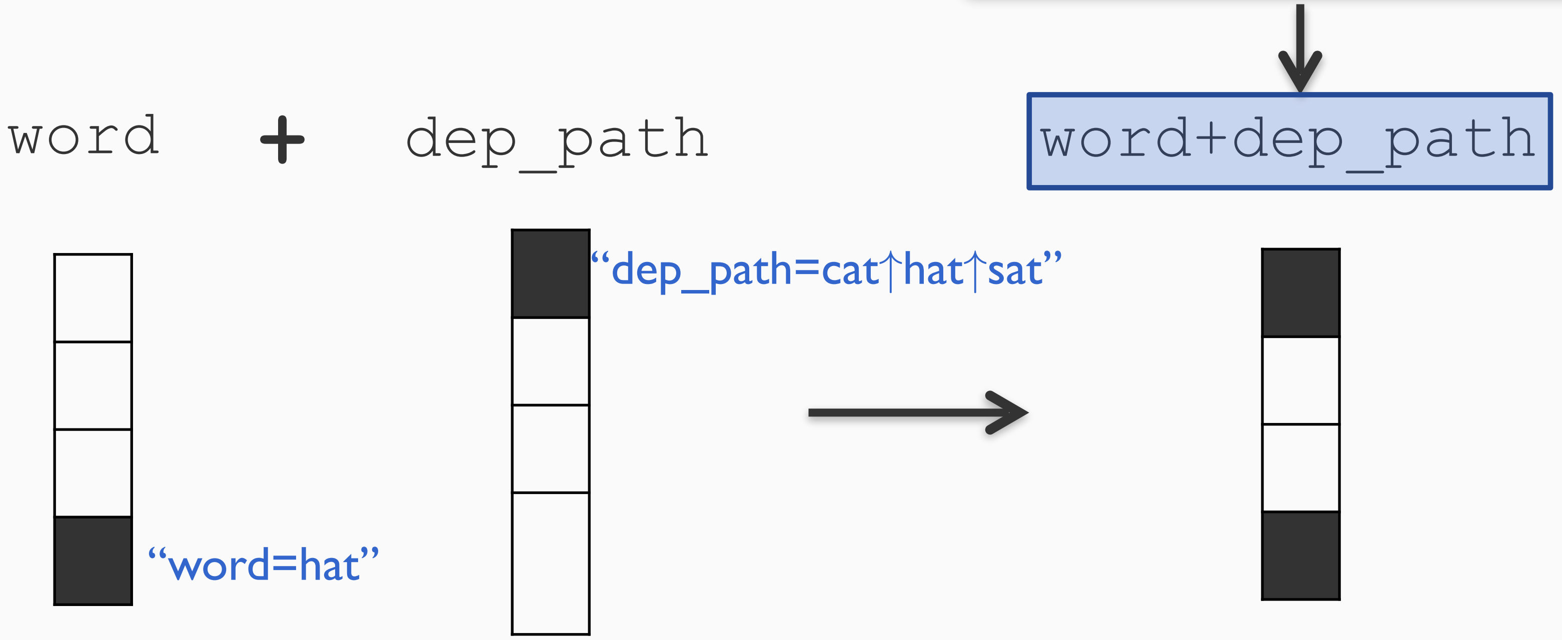
Feature addition



Operations on these functions

Feature addition

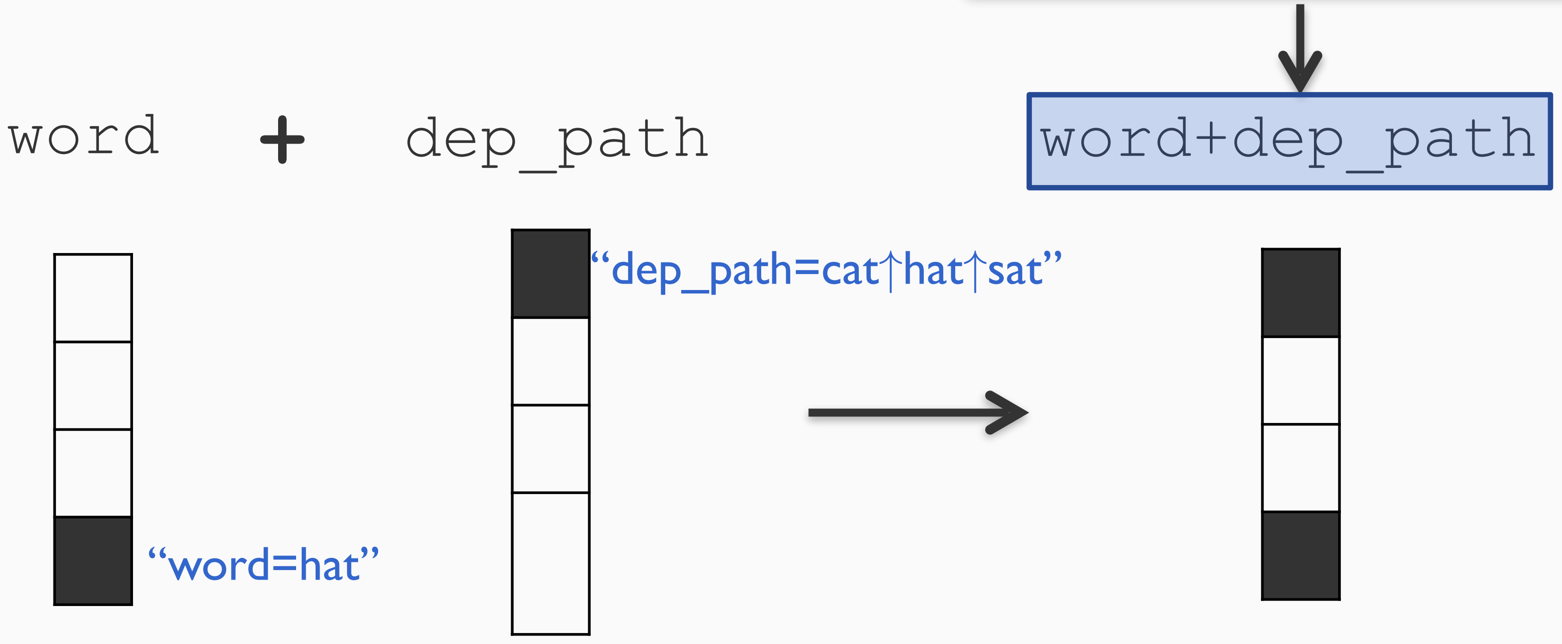
This is a function that maps inputs to a vector space
i.e, it is a feature extractor



Operations on these functions

Feature addition

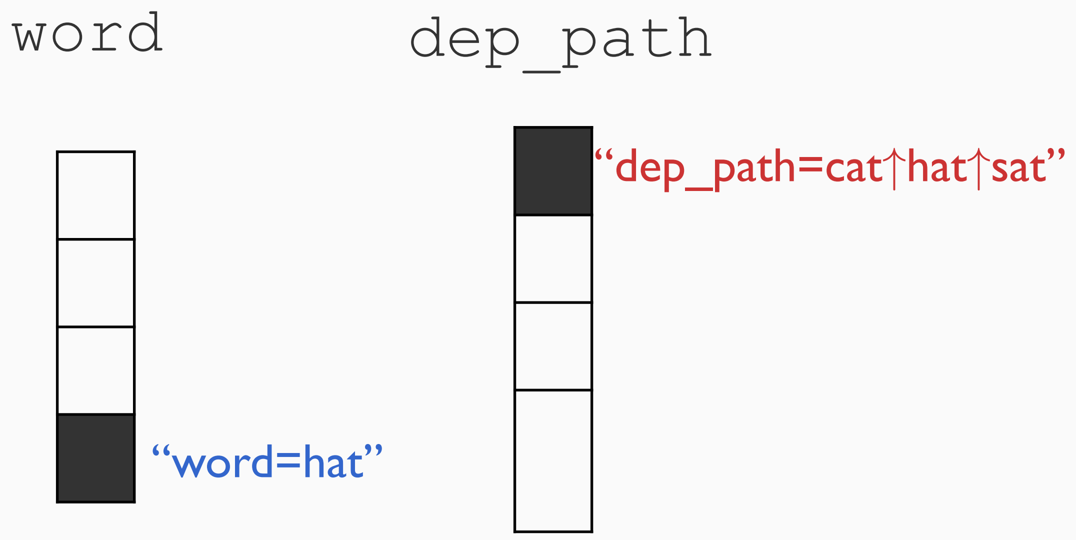
This is a function that maps inputs to a vector space
i.e, it is a feature extractor



F = Set of all feature extractors

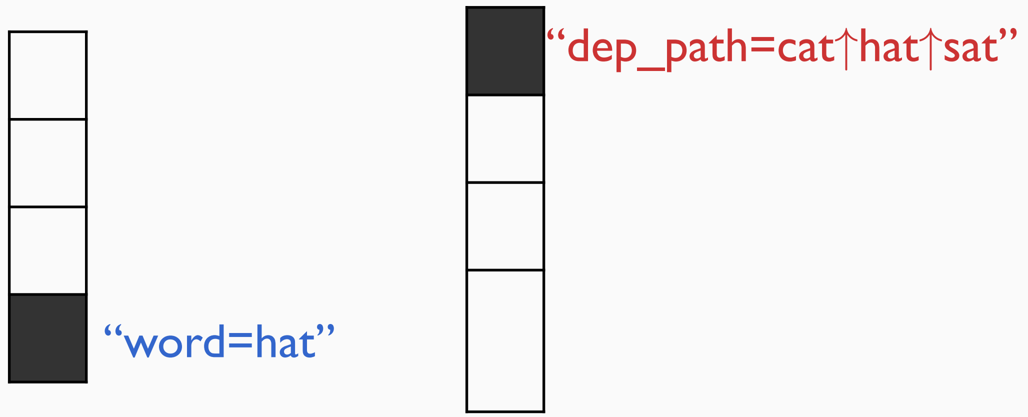
$+ : F \times F \rightarrow F$

Feature conjunctions



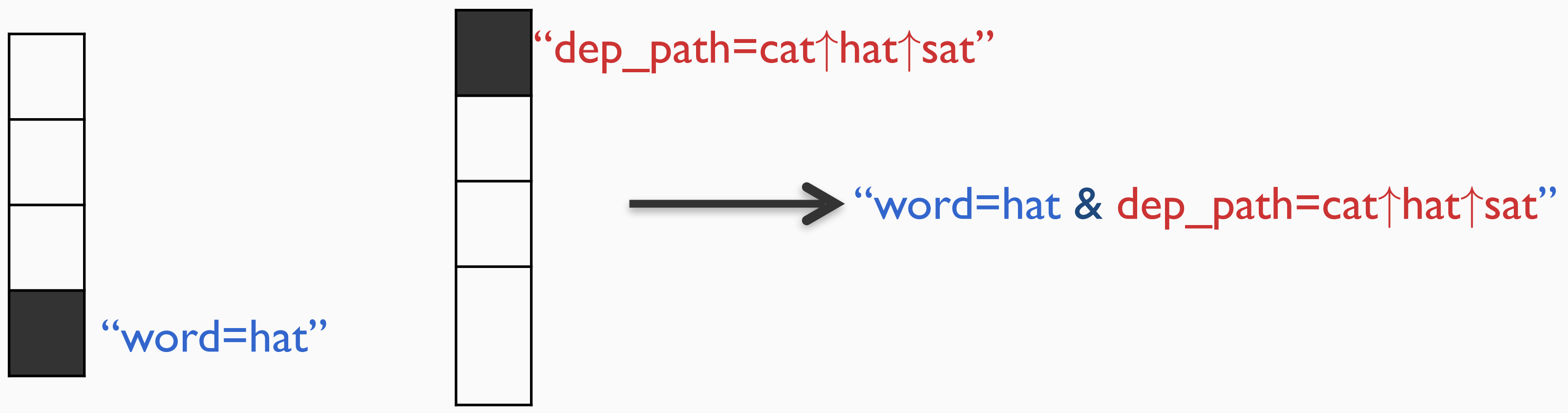
Feature conjunctions

`word` & `dep_path`



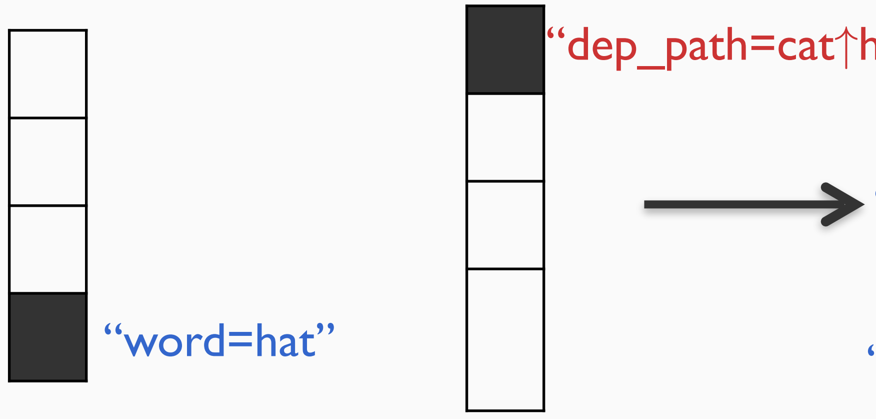
Feature conjunctions

`word` **&** `dep_path`



Feature conjunctions

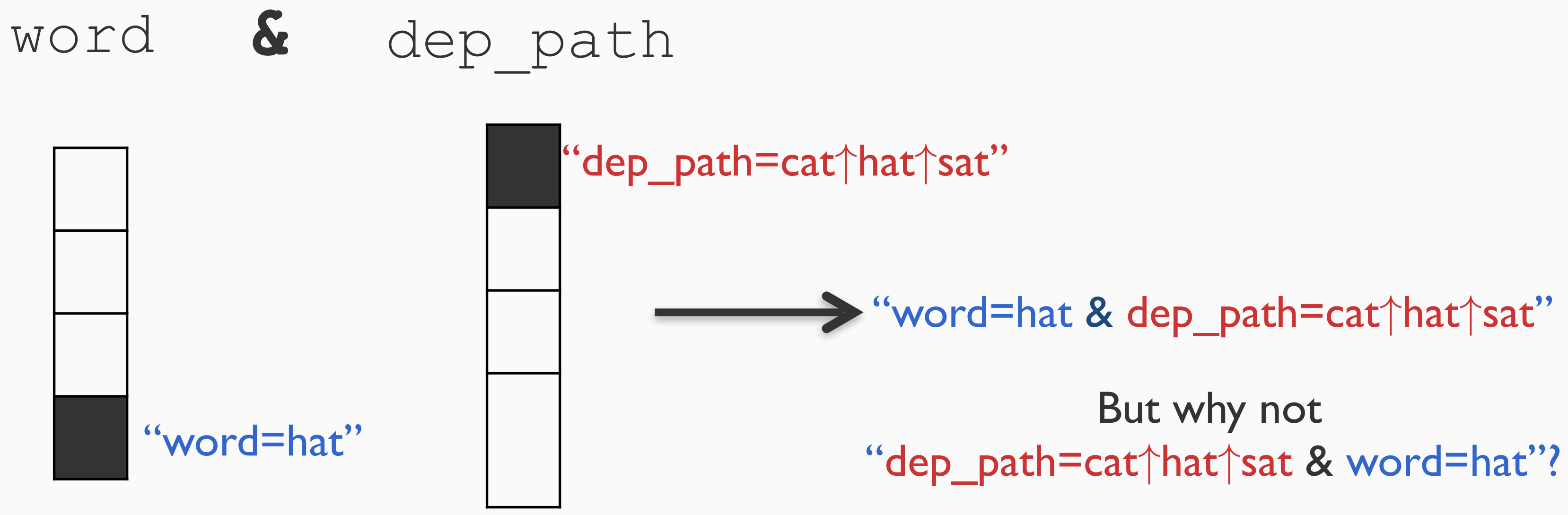
`word` & `dep_path`



“word=hat & dep_path=cat↑hat↑sat”

But why not
“dep_path=cat↑hat↑sat & word=hat”?

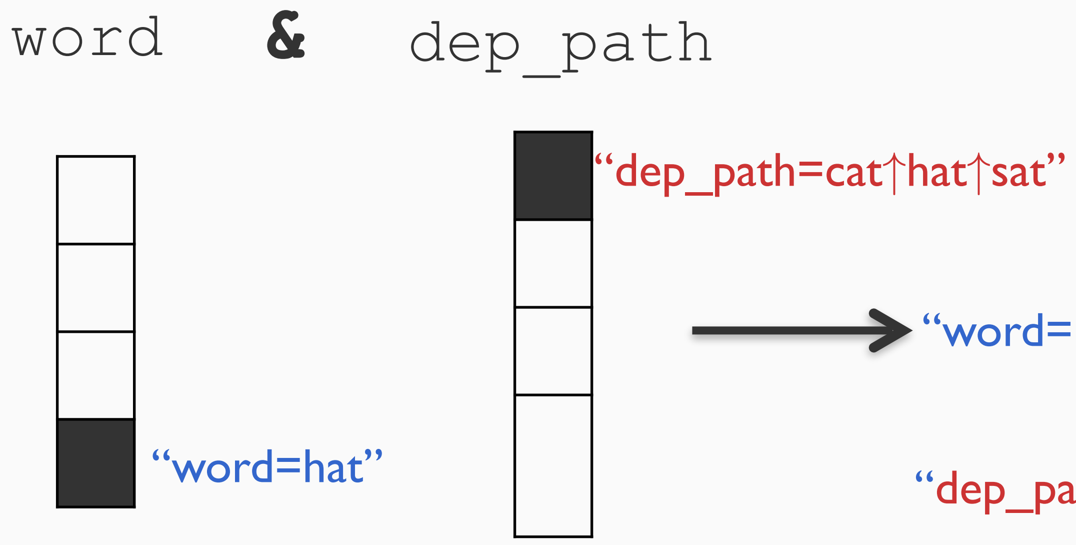
Feature conjunctions



But why not
"dep_path=cat↑hat↑sat & word=hat"?

Feature conjunctions equivalent to symmetric tensor products.
Captures the idea that conjunctions are symmetric

Feature conjunctions



→ "word=hat & dep_path=cat↑hat↑sat"

But why not
"dep_path=cat↑hat↑sat & word=hat"?

Feature conjunctions equivalent to symmetric tensor products.
Captures the idea that conjunctions are symmetric

$\mathbf{F}$ = Set of all feature extractors

$\&: F \times F \rightarrow F$

Formalizing Feature Extraction

Feature extractors Functions from any objects to a vector space. Examples: `word`, `pos`, `dep_path`, etc.

Formalizing Feature Extraction

Feature extractors Functions from any objects to a vector space. Examples: `word`, `pos`, `dep_path`, etc.

Special feature extractors

- `0`: The **`zero`** feature extractor, maps inputs to the zero vector.
- `1`: The **`bias`** feature extractor, maps inputs to a fixed bias feature vector.

Formalizing Feature Extraction

Feature extractors Functions from any objects to a vector space. Examples: `word`, `pos`, `dep_path`, etc.

Special feature extractors

- `0`: The **`zero`** feature extractor, maps inputs to the zero vector.
- `1`: The **`bias`** feature extractor, maps inputs to a fixed bias feature vector.

Operators on feature extractors Compose feature functions to produce new feature extractors.

Formalizing Feature Extraction

Feature extractors Functions from any objects to a vector space. Examples: `word`, `pos`, `dep_path`, etc.

Special feature extractors

- `0`: The **`zero`** feature extractor, maps inputs to the zero vector.
- `1`: The **`bias`** feature extractor, maps inputs to a fixed bias feature vector.

Operators on feature extractors Compose feature functions to produce new feature extractors.

Feature addition:
(eg: `word+pos`)
Add the vectors produced by the operands

Feature conjunction:
(eg: `word&pos`)
Symmetric tensor product of the vectors produced by the operands

These definitions align with the intuitive interpretations of these operators.

An algebra for feature extraction

Theorem: The set of feature extractors, with feature addition and conjunction, forms a **commutative semiring**.

An algebra for feature extraction

Theorem: The set of feature extractors, with feature addition and conjunction, forms a **commutative semiring**.

1. Feature addition is

- Associative
  `(a+b)+c = a+(b+c)`
- Commutative `a+b = b+a`
- **`zero`** is the identity element

An algebra for feature extraction

Theorem: The set of feature extractors, with feature addition and conjunction, forms a **commutative semiring**.

1. Feature addition is
- Associative
  `(a+b)+c = a+(b+c)`
- Commutative `a+b = b+a`
- **`zero`** is the identity element

2. Feature conjunction is
- Associative
  `(a&b)&c = a&(b&c)`
- Commutative `a&b = b&a`
- **`bias`** is the identity element

An algebra for feature extraction

Theorem: The set of feature extractors, with feature addition and conjunction, forms a **commutative semiring**.

1. Feature addition is
- Associative
  `(a+b)+c = a+(b+c)`
- Commutative `a+b = b+a`
- **`zero`** is the identity element

2. Feature conjunction is
- Associative
  `(a&b)&c = a&(b&c)`
- Commutative `a&b = b&a`
- **`bias`** is the identity element

3. Multiplication distributes over addition
`a&(b+c) = a&b+a&c`

An algebra for feature extraction

Theorem: The set of feature extractors, with feature addition and conjunction, forms a **commutative semiring**.

1. Feature addition is
   - Associative
     `(a+b)+c` = `a+(b+c)`
   - Commutative `a+b` = `b+a`
   - **`zero`** is the identity element

2. Feature conjunction is
   - Associative
     `(a&b)&c` = `a&(b&c)`
   - Commutative `a&b` = `b&a`
   - **`bias`** is the identity element

3. Multiplication distributes over addition
   `a&(b+c)` = `a&b+a&c`

4. Conjoining with the zero feature extractor gives back zero

An algebra for feature extraction

Theorem: The set of feature extractors, with feature addition and conjunction, forms a **commutative semiring**.

1. Feature addition is
   - Associative `(a+b)+c = a+(b+c)`
   - Commutative `a+b = b+a`
   - **`zero`** is the identity element

2. Feature conjunction is
   - Associative `(a&b)&c = a&(b&c)`
   - Commutative `a&b = b&a`
   - **`bias`**is the identity element

3. Multiplication distributes over addition
`a&(b+c) = a&b+a&c`

4. Conjoining with the zero feature extractor gives back zero

An algebra for feature extraction

Theorem: The set of feature extractors, with feature addition and conjunction, forms a **commutative semiring**.

1. Feature addition is
- Associative (a+b
  (b+c)
- Commutative a+
- `zero` is the ident

2. Feature conjunction is

&c =

= b&a

element

So what?

3. Multiplication distributes over addition
`a&(b+c) = a&b+a&c`

4. Conjoining with the zero feature extractor gives back zero

An opportunity for speedup

Apply distributive property to refactor feature extractors

`a&b+a&c`

Two conjunctions
One addition

An opportunity for speedup

Apply distributive property to refactor feature extractors

```
a&b+a&c = a&(b+c)
```

Two conjunctions	***One*** conjunction
One addition	One addition

An opportunity for speedup

Apply distributive property to refactor feature extractors

```
a&b+a&c = a&(b+c)
```

Two conjunctions One addition	***One*** conjunction One addition

Looks trivial. But saves computation in two ways:

1. Fewer conjunctions
2. If done carefully, we can automatically move expensive feature extractors outside

And this is done at a symbolic level before any feature extractor is ever applied

An opportunity for speedup

Apply distributive property to refactor feature extractors

`a&b+a&c = a&(b+c)`

Two conjunctions One addition	***One*** conjunction One addition

Looks trivial. But saves computation in two ways:

1. Fewer conjunctions
2. If done carefully, we can automatically move expensive feature extractors outside

And this is done at a symbolic level before any feature extractor is ever applied

Can we do this systematically?

From algebra to an algorithm

Commutative semirings admit the use of the **Generalized Distributive Law (GDL) algorithm** to calculate sums of products faster.

A message passing algorithm that takes familiar forms for various semirings.

Eg: Viterbi, Baum-Welch, Belief propagation algorithms.

Aji, Srinivas M., and Robert J. McEliece. "The generalized distributive law." *IEEE transactions on Information Theory* 46.2 (2000): 325-343.

w: word, p: pos, d: dep-path

Making Feature Extraction Faster

```
f = w + p + d + w&d + p&d
```

`w`: `word`, `p`: `pos`, `d`: `dep-path`

Making Feature Extraction Faster

$$f = w + p + d + w\&d + p\&d$$

1. Convert any feature extractor into a canonical sum of products form

$$f = w\&\mathbb{1}\&\mathbb{1} + \mathbb{1}\&p\&\mathbb{1} + \mathbb{1}\&\mathbb{1}\&d + w\&\mathbb{1}\&d + \mathbb{1}\&p\&d$$

Now the goal is compute this sum of products efficiently.

w: `word`, p: `pos`, d: `dep-path`

Making Feature Extraction Faster

$$f = w + p + d + w\&d + p\&d$$

1. Convert any feature extractor into a canonical sum of products form

$$f = w\&\mathbb{1}\&\mathbb{1} + \mathbb{1}\&p\&\mathbb{1} + \mathbb{1}\&\mathbb{1}\&d + w\&\mathbb{1}\&d + \mathbb{1}\&p\&d$$

Now the goal is compute this sum of products efficiently.

2. Construct a junction tree and assign local potential functions to each node

`w: word, p: pos, d: dep-path`

Making Feature Extraction Faster

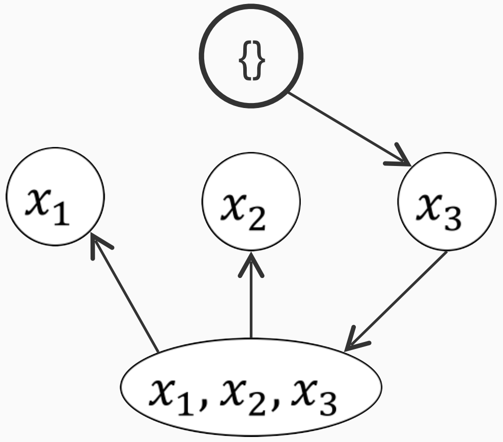
`f = w + p + d + w&d + p&d`

1. Convert any feature extractor into a canonical sum of products form

`f = w&𝟙&𝟙 + 𝟙&p&𝟙 + 𝟙&𝟙&d`
`+ w&𝟙&d + 𝟙&p&d`

Now the goal is compute this sum of products efficiently.

2. Construct a junction tree and assign local potential functions to each node



w: word, p: pos, d: dep-path

Making Feature Extraction Faster

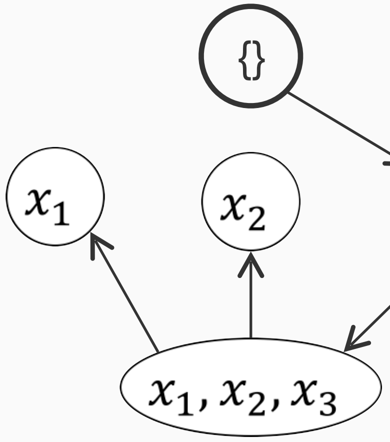
`f = w + p + d + w&d + p&d`

1. Convert any feature extractor into a canonical sum of products form

`f = w&𝟙&𝟙 + 𝟙&p&𝟙 + 𝟙&𝟙&d + w&𝟙&d + 𝟙&p&d`

Now the goal is compute this sum of products efficiently.

2. Construct a junction tree and assign local potential functions to each node



3. Run message passing from leaves to root (using the semiring operators)

Message at root:

`w + p + d&(𝟙+w+p)`

`w: word, p: pos, d: dep-path`

Making Feature Extraction Faster

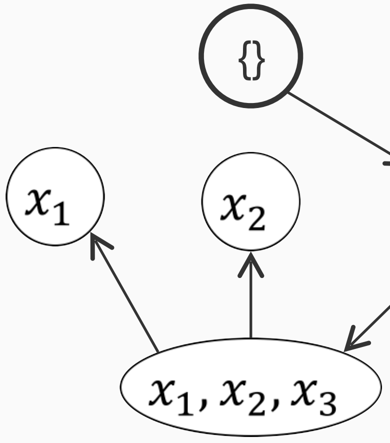
`f = w + p + d + w&d + p&d`

1. Convert any feature extractor into a canonical sum of products form

`f = w&𝟙&𝟙 + 𝟙&p&𝟙 + 𝟙&𝟙&d + w&𝟙&d + 𝟙&p&d`

Now the goal is compute this sum of products efficiently.

2. Construct a junction tree and assign local potential functions to each node



3. Run message passing from leaves to root (using the semiring operators)

Message at root:

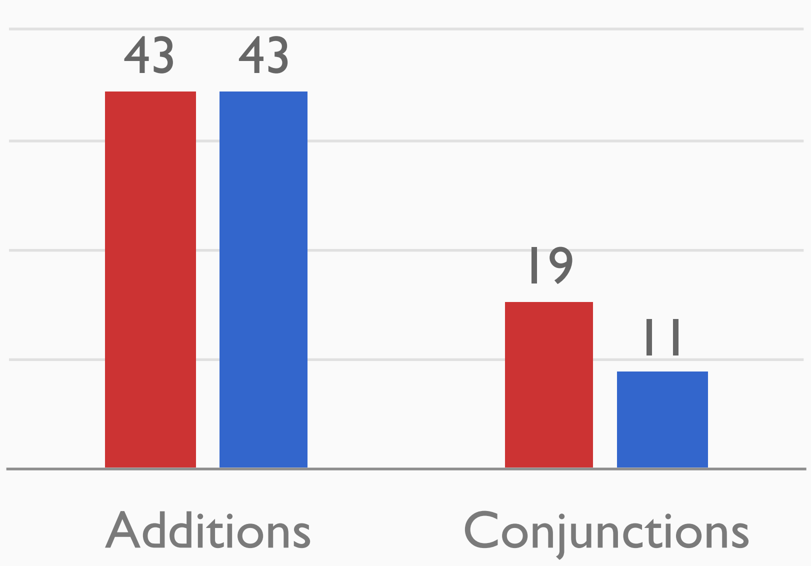
`w + p + d&(𝟙+w+p)`

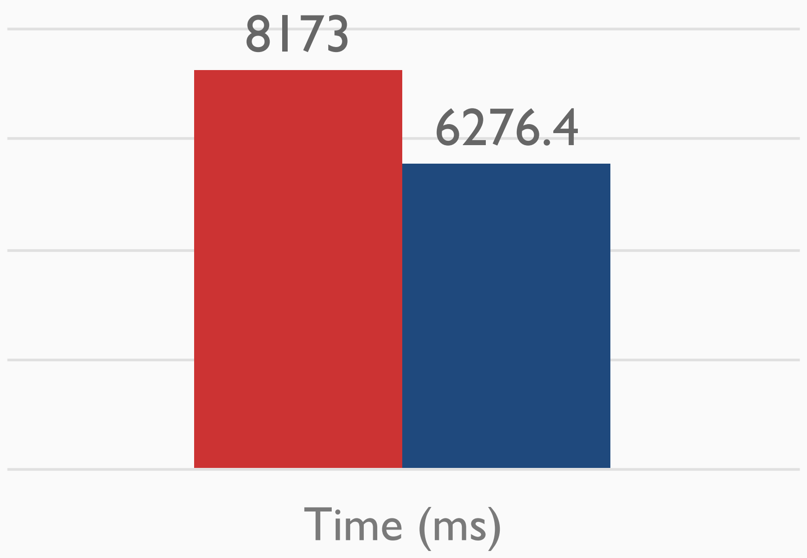
Functionally equivalent factorized feature extractor. One less conjunction, and dependency path is computed only once.

1. ACE relation extraction

Assigning a relation label to a pair of entities. Feature set from Zhou et al 2005

🟥 Original 🟦 Factorized





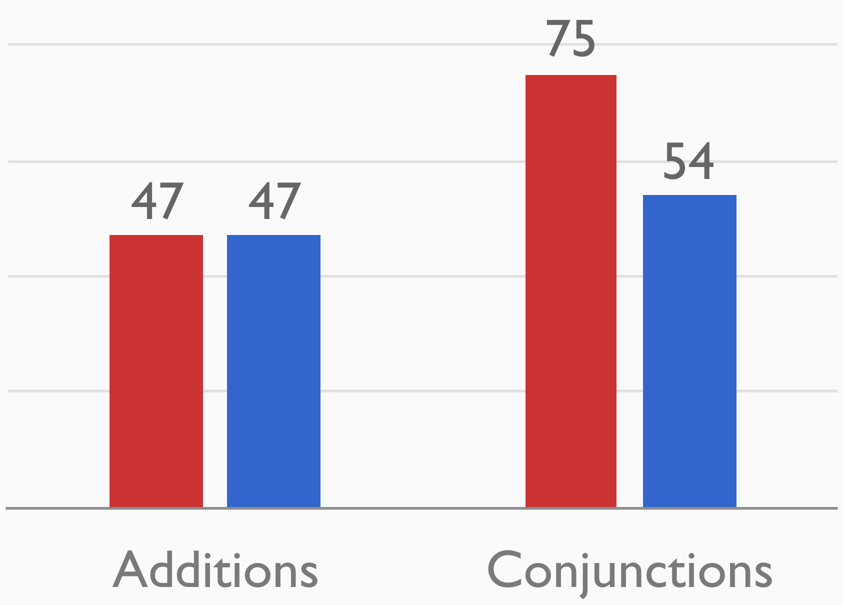
1. Refactoring decreases number of conjunctions (at template level)

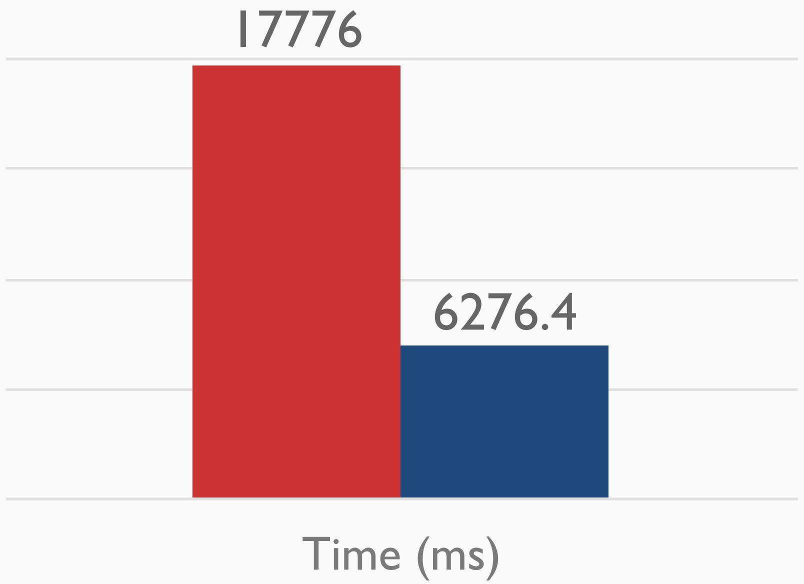
2. Wall clock time improvements. (Time in ms averaged multiple runs over the dataset.)

2. Text Chunking

CoNLL text chunking task. Feature set from Martins et al 2011

🟥 Original 🟦 Factorized





1. Refactoring decreases number of conjunctions (at template level)
2. Wall clock time improvements. (Time in ms averaged multiple runs over the dataset.)

An algebra for feature extraction

$$\begin{tikzcd}
\text{A formalization of feature extraction} \arrow[r] & \text{An algebra of feature extractors} \arrow[r] & \text{A message passing algorithm that makes feature extraction faster}
\end{tikzcd}$$

The punchline: A way to automatically refactor feature extractors to be faster

A ground up rethinking

- Faster inference [Srikumar et al 2012, Kundu et al 2013]
  - No need to solve certain inference problems if they are similar to ones we have already solved
- Faster feature extraction [Srikumar 2017]
  - Automatically refactor feature extraction to reduce redundant computation
- Faster dot products [Shafiee et al 2016]
  - Better machine level support, novel hardware

Challenges

- Representations

  Linguistic intuitions can help develop useful *structured* representations for performing reasoning about text

- Data

  Exploit the need for consistency between different kinds of linguistic predictions to act as a surrogate for training data

- Efficiency
  - Predictive models need to scale (eg: to every doctor and patient in the world)

Challenges

- Representations

  Linguistic intuitions can help develop useful *structured* representations for performing reasoning about text

- Data

  Exploit the need for consistency between different kinds of linguistic predictions to act as a surrogate for training data

- Efficiency

  Do not perform any redundant computation by automatically discovering regularities

This talk

Natural language processing has the tremendous opportunity to revolutionize many fields,

…but we need to overcome several challenges.

- Empathy and the Machine: A case study of NLP and Mental Health

- Challenge 1: The big [data problem]

- Challenge 2: Scaling NLP for everyone

This talk

Natural language processing has the tremendous opportunity to revolutionize many fields,

…but we need to overcome several challenges.

- Empathy and the Machine: A case study of NLP and Mental Health

- Challenge 1: The big [data problem]

- Challenge 2: Scaling NLP for everyone

Questions?