Acquiring Lexical Semantic Knowledge

Vered Shwartz

Talk at NUS School of Computing, January 9, 2018

B I U
N L P

Outline

Introduction and Motivation

Recognizing Semantic Relations between Nouns

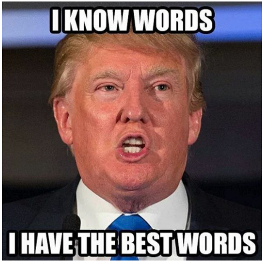
Hypernymy Detection

Other Semantic Relations

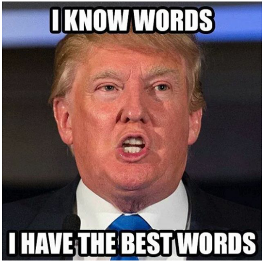
Interpreting Noun-Compounds

Acquiring Predicate Paraphrases

What is "lexical knowledge"?

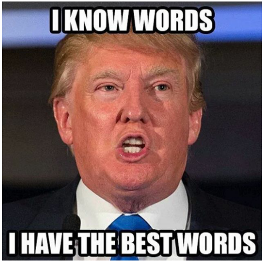


What is “lexical knowledge”?



- Knowledge about how words **relate** to each other.

What is "lexical knowledge"?



- Knowledge about how words **relate** to each other.

- Various relations are helpful for dealing with lexical variability:
  - synonymy (*elevator*, *lift*)
  - hypernymy (*dog*, *pet*)
  - part-of (*Tel Aviv*, *Israel*)
  - more...

Example Application 1 - Search

Query

"Actors engaged in Scientology"

Results

Tom Cruise must ditch the vile cult of Scientology NOW before
www.dailymail.co.uk/.../PIERS-MORGAN-**Tom-Cruise**-ditch-vile-cult-**Scie**...
Apr 2, 2015 - PIERS MORGAN: I wanted to be **Tom Cruise**. As a fresh-faced, 21-year-old, I watched Top Gun a dozen times at my local movie theater in ...

John Travolta Says Scientology Is A Target Because It ...
www.huffingtonpost.com/.../**john-travolta-scientology**-target_n_710268...
Apr 20, 2015 - During an interview with "Good Morning America" Monday, **John Travolta** was asked why there is so much intrigue and interest surrounding the ...

Knowledge

Tom Cruise and *John Travolta* are instances of *actor*.

Example Application 2 - Question Answering

Question

"When did Donald Trump visit in Alabama?"

Candidate Passages

1. Trump visited Huntsville on September 23.
2. Trump visited Mississippi on June 21.

Knowledge

Huntsville is in Alabama, but *Mississippi* is not.

Word Embeddings

First, let's get this off the table: ***"why not just use word embeddings?"***

Word Embeddings

First, let's get this off the table: ***"why not just use word embeddings?"***

- Word embeddings are great in capturing semantic relatedness!

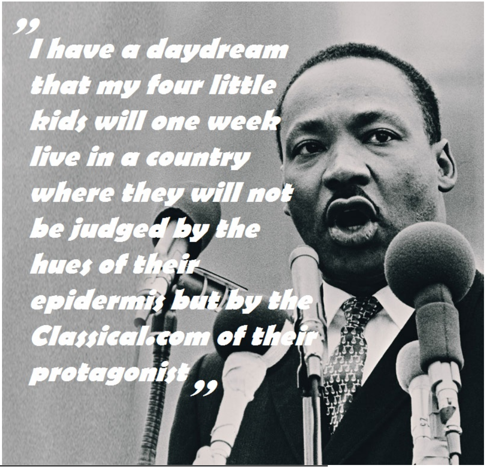
Word Embeddings

First, let's get this off the table: ***"why not just use word embeddings?"***

- Word embeddings are great in capturing semantic relatedness!
- ...but they mix all semantic relations together.

Word Embeddings

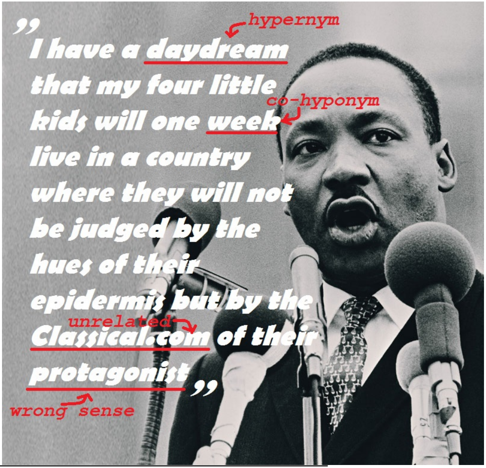
- To illustrate, take famous texts and replace nouns with their word2vec neighbours:[1]



[1] More examples here: `https://goo.gl/LJHzbi`

Word Embeddings

- To illustrate, take famous texts and replace nouns with their word2vec neighbours:[1]



[1] More examples here: `https://goo.gl/LJHzbi`

Recognizing Semantic Relations between Nouns

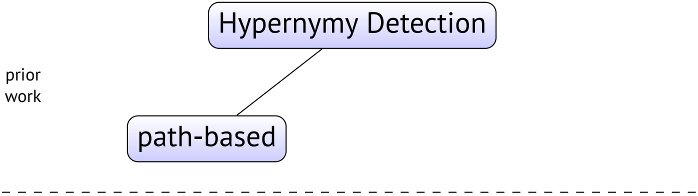
The Hypernymy Detection Task

- We first focused on hypernymy
  - The hyponym is a subclass of / instance of the hypernym
  - *(cat, animal)*, *(Google, company)*

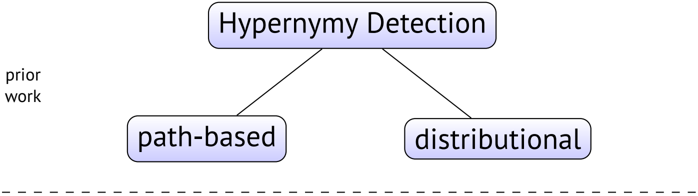
The Hypernymy Detection Task

- We first focused on hypernymy
  - The hyponym is a subclass of / instance of the hypernym
  - *(cat, animal)*, *(Google, company)*
- Given two terms, x and y, decide whether y is a hypernym of x
  - in some senses of x and y, e.g. *(apple, fruit), (apple, company)*

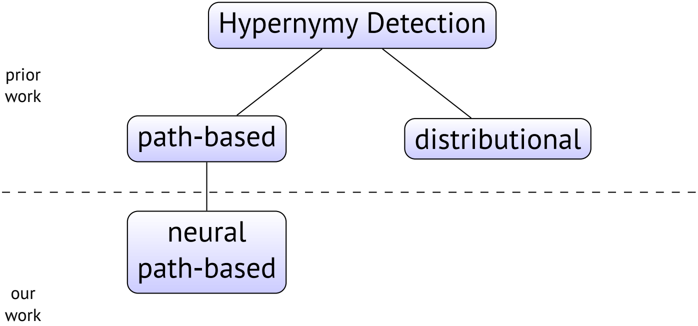
Corpus-based Hypernymy Detection



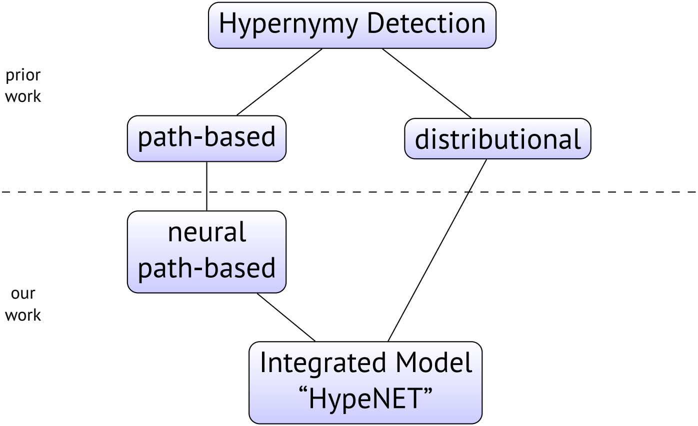
Corpus-based Hypernymy Detection



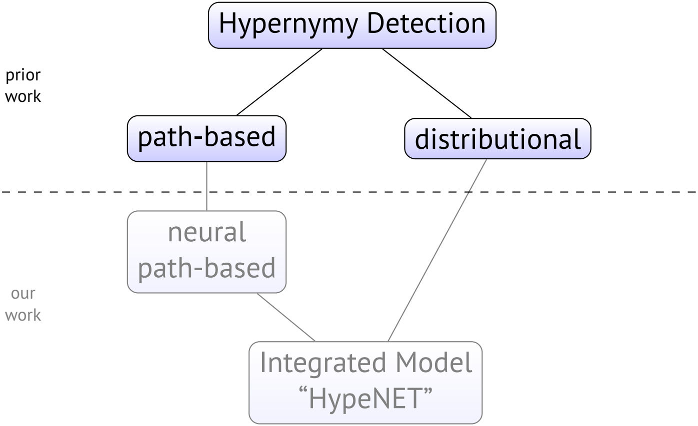
Corpus-based Hypernymy Detection



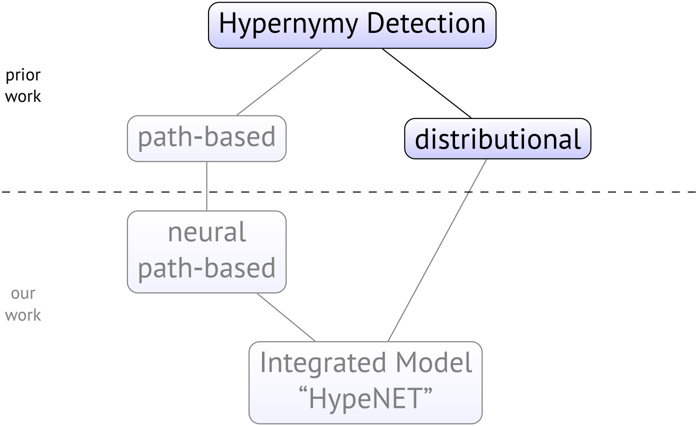
Corpus-based Hypernymy Detection



Prior Methods



Distributional Approach



Distributional Approach

- Recognize the relation between words based on their *separate* occurrences in the corpus

Distributional Approach

- Recognize the relation between words based on their *separate* occurrences in the corpus

- Distributional Hypothesis [Harris, 1954]:
  Words that occur in similar contexts tend to have similar meanings

Distributional Approach

- Recognize the relation between words based on their *separate* occurrences in the corpus
- Distributional Hypothesis [Harris, 1954]:
  Words that occur in similar contexts tend to have similar meanings
  - e.g. *elevator* and *lift* will both appear next to *up*, *floor* and *stairs*

Distributional Approach

- Recognize the relation between words based on their *separate* occurrences in the corpus

- Distributional Hypothesis [Harris, 1954]:
  Words that occur in similar contexts tend to have similar meanings
  - e.g. *elevator* and *lift* will both appear next to *up*, *floor* and *stairs*

- Word embeddings [Mikolov et al., 2013, Pennington et al., 2014] are low-dimensional vector representations of words
  - Similar words have similar vectors

Supervised Distributional Methods

- Represent (x, y) as a feature vector, based of the terms' embeddings:
  - Concatenation $\vec{x} \oplus \vec{y}$ [Baroni et al., 2012]
  - Difference $\vec{y} - \vec{x}$ [Roller et al., 2014, Weeds et al., 2014]

Supervised Distributional Methods

- Represent (x, y) as a feature vector, based of the terms' embeddings:
  - Concatenation $\vec{x} \oplus \vec{y}$ [Baroni et al., 2012]
  - Difference $\vec{y} - \vec{x}$ [Roller et al., 2014, Weeds et al., 2014]
- Train a classifier to predict whether y is a hypernym of x

Supervised Distributional Methods

- Represent (x, y) as a feature vector, based of the terms' embeddings:
  - Concatenation $\vec{x} \oplus \vec{y}$ [Baroni et al., 2012]
  - Difference $\vec{y} - \vec{x}$ [Roller et al., 2014, Weeds et al., 2014]
- Train a classifier to predict whether y is a hypernym of x

- Achieved very good results on common hypernymy detection datasets

Supervised Distributional Methods

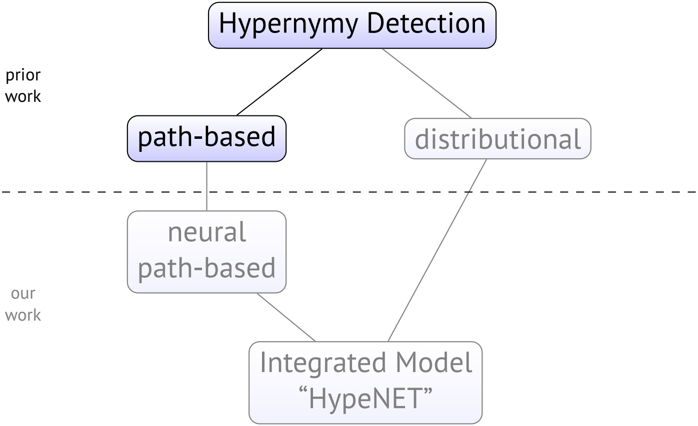
- Represent (x, y) as a feature vector, based of the terms' embeddings:
  - Concatenation $\vec{x} \oplus \vec{y}$ [Baroni et al., 2012]
  - Difference $\vec{y} - \vec{x}$ [Roller et al., 2014, Weeds et al., 2014]
- Train a classifier to predict whether y is a hypernym of x
- Achieved very good results on common hypernymy detection datasets
- Is it a solved task?

Supervised Distributional Methods

- Represent (x, y) as a feature vector, based of the terms' embeddings:
  - Concatenation $\vec{x} \oplus \vec{y}$ [Baroni et al., 2012]
  - Difference $\vec{y} - \vec{x}$ [Roller et al., 2014, Weeds et al., 2014]
- Train a classifier to predict whether y is a hypernym of x

- Achieved very good results on common hypernymy detection datasets
- Is it a solved task?
- Probably not. They don't learn the *relation* between x and y, but mostly that y is a *prototypical hypernym* [Levy et al., 2015].
  - e.g. that *(x, fruit)* or *(x, animal)* are always hypernyms

Path-based Approach



Path-based Approach

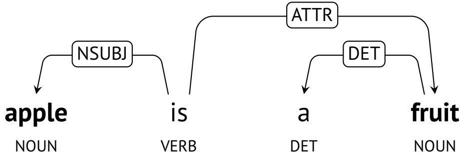
- Recognize the relation between *x* and *y* based on their *joint* occurrences in the corpus

Path-based Approach

- Recognize the relation between *x* and *y* based on their *joint* occurrences in the corpus
- Hearst Patterns [Hearst, 1992] - patterns connecting *x* and *y* may indicate that *y* is a hypernym of *x*
    - e.g. *X or other Y*, *X is a Y*, *Y, including X*

Path-based Approach

- Recognize the relation between *x* and *y* based on their *joint* occurrences in the corpus
- Hearst Patterns [Hearst, 1992] - patterns connecting *x* and *y* may indicate that *y* is a hypernym of *x*
  - e.g. *X or other Y*, *X is a Y*, *Y, including X*
- Patterns can be represented using dependency paths:



Supervised Path-based Approach

- Supervised method to recognize hypernymy [Snow et al., 2004]:

Supervised Path-based Approach

- Supervised method to recognize hypernymy [Snow et al., 2004]:
  - Features: all dependency paths that connected *x* and *y* in a corpus:

0	0	...	58	0	...	97	0	...	0
			↑ X and other Y			↑ such Y as X			

Supervised Path-based Approach

- Supervised method to recognize hypernymy [Snow et al., 2004]:
  - Features: all dependency paths that connected *x* and *y* in a corpus:

0	0	...	58	0	...	97	0	...	0
			↑ X and other Y			↑ such Y as X			

  - Trained a logistic regression classifier to predict hypernymy

Path-based Approach Issues

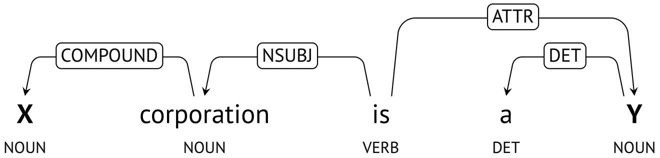
- The feature space is too sparse:

Path-based Approach Issues

- The feature space is too sparse:
  - Similar paths share no information:
    X inc. is a Y
    X group is a Y
    X organization is a Y

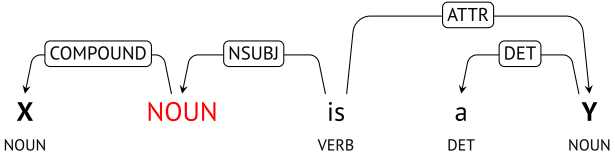
Path-based Approach Issues

- The feature space is too sparse:
  - Similar paths share no information:
    X inc. is a Y
    X group is a Y
    X organization is a Y
- PATTY [Nakashole et al., 2012] generalized paths, by replacing a word by:



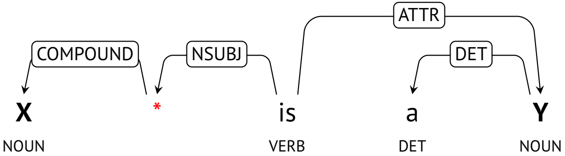
Path-based Approach Issues

- The feature space is too sparse:
  - Similar paths share no information:
    X inc. is a Y
    X group is a Y
    X organization is a Y
- PATTY [Nakashole et al., 2012] generalized paths, by replacing a word by:
  - its POS tag



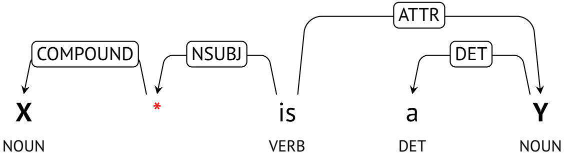
Path-based Approach Issues

- The feature space is too sparse:
  - Similar paths share no information:
    X inc. is a Y
    X group is a Y
    X organization is a Y
- PATTY [Nakashole et al., 2012] generalized paths, by replacing a word by:
  - a wild-card



Path-based Approach Issues

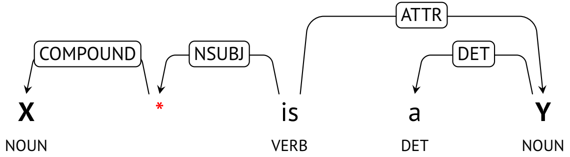
- The feature space is too sparse:
  - Similar paths share no information:
    X inc. is a Y
    X group is a Y
    X organization is a Y
- PATTY [Nakashole et al., 2012] generalized paths, by replacing a word by:
  - a wild-card



- Some of these generalizations are too general:
  - X is defined as Y ≈ X is described as Y via X is VERB as Y

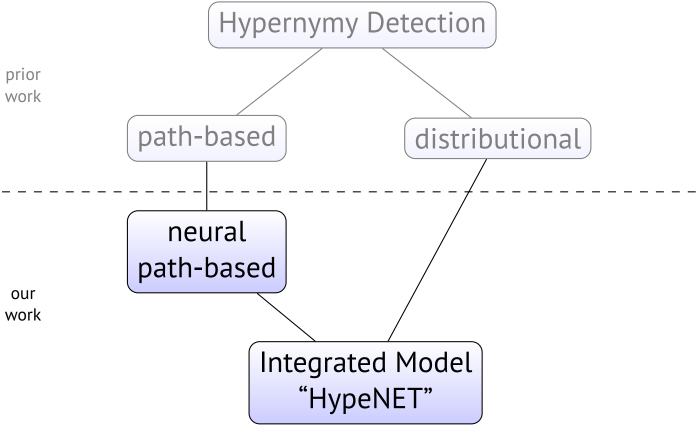
Path-based Approach Issues

- The feature space is too sparse:
  - Similar paths share no information:
    X inc. is a Y
    X group is a Y
    X organization is a Y
- PATTY [Nakashole et al., 2012] generalized paths, by replacing a word by:
  - a wild-card

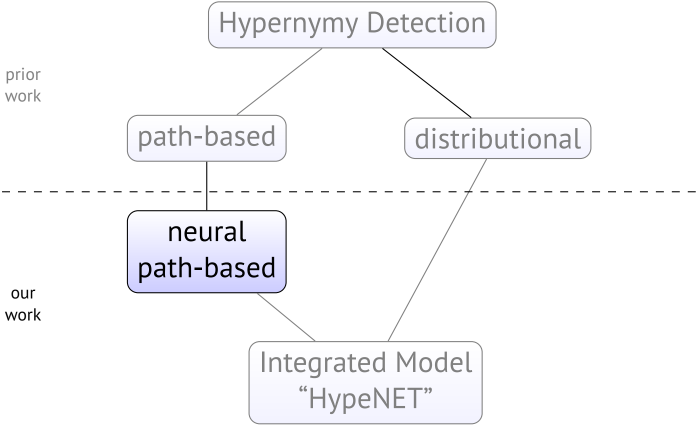


- Some of these generalizations are too general:
  - X is defined as Y $\approx$ X is described as Y via X is VERB as Y
  - X is defined as Y $\neq$ X is rejected as Y

HypeNET: Integrated Path-based and Distributional Method [Shwartz et al., 2016]

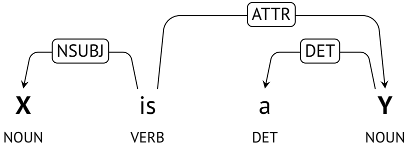


First Step: Improving Path Representation



Path Representation (1/2)

1. Split each path to edges



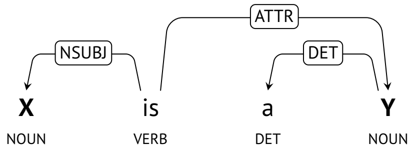
X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/<

- Each edge consists of 4 components:

dependent lemma / dependent POS / dependency label / direction

Path Representation (1/2)

1. Split each path to edges



`X/NOUN/nsubj/>`   `be/VERB/ROOT/-`   `Y/NOUN/attr/<`

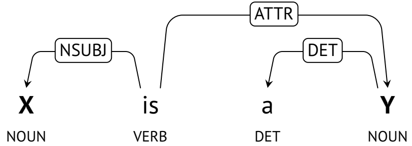
- Each edge consists of 4 components:

  `dependent lemma` / `dependent POS` / `dependency label` / `direction`

- We learn embedding vectors for each component
  - Lemma: initialized with pre-trained word embeddings

Path Representation (1/2)

1. Split each path to edges



X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/<

- Each edge consists of 4 components:

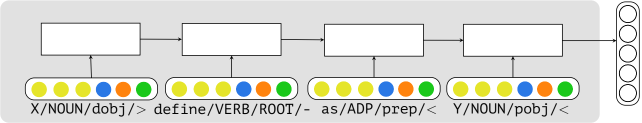
  `dependent lemma` / `dependent POS` / `dependency label` / `direction`

- We learn embedding vectors for each component
  - Lemma: initialized with pre-trained word embeddings
- The edge's vector is the concatenation of its components' vectors:

be/VERB/ROOT/-

Path Representation (2/2)

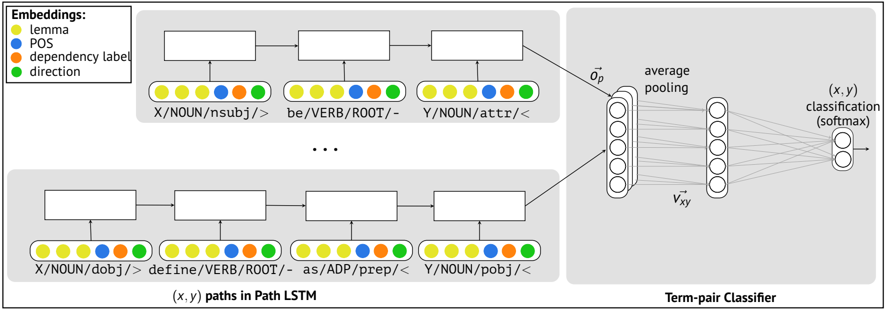
2. Feed the edges sequentially to an LSTM



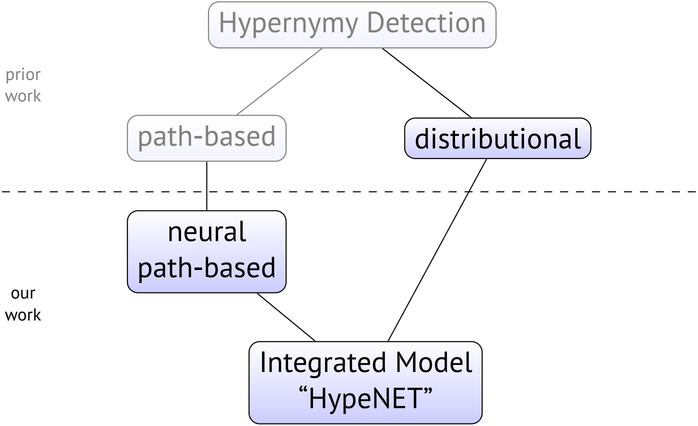
- Use the last output vector as the path embedding
- The LSTM may focus on edges that are more informative for the classification task, while ignoring others

Term-pair Classification

- The LSTM encodes a single path
- Each term-pair has multiple paths
  - Represent a term-pair as its averaged path embedding
- Classify for hypernymy (path-based network):



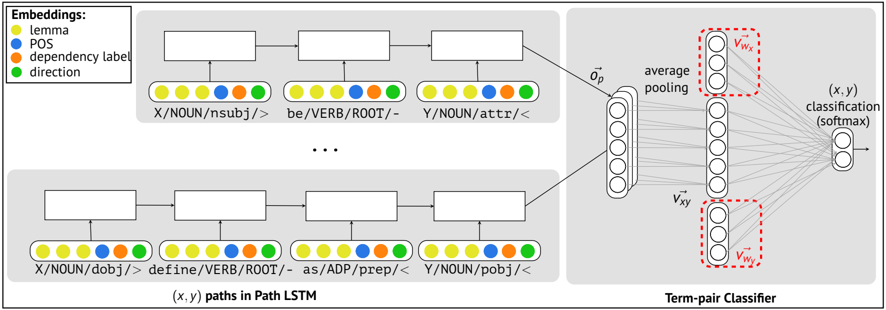
Second Step: Integrating Distributional Information



Second Step: Integrating Distributional Information

- Integrated network: add distributional information
  - Simply concatenate *x* and *y*'s word embeddings to the averaged path

- Classify for hypernymy (integrated network):



Results

- On a new dataset, built from knowledge resources

method		precision	recall	F_1
Path-based	Snow	0.843	0.452	0.589
	Snow + GEN	0.852	0.561	0.676
	HypeNET Path-based	0.811	0.716	0.761
Distributional	Best Supervised	0.901	0.637	0.746
Integrated	HypeNET Integrated	**0.913**	**0.890**	**0.901**

- Path-based:
  - Compared to Snow + Snow with PATTY style generalizations
  - Our method outperforms path-based baselines with improved recall

Results

- On a new dataset, built from knowledge resources

method		precision	recall	F_1
Path-based	Snow	0.843	0.452	0.589
	Snow + GEN	0.852	0.561	0.676
	HypeNET Path-based	0.811	0.716	0.761
Distributional	Best Supervised	0.901	0.637	0.746
Integrated	HypeNET Integrated	**0.913**	**0.890**	**0.901**

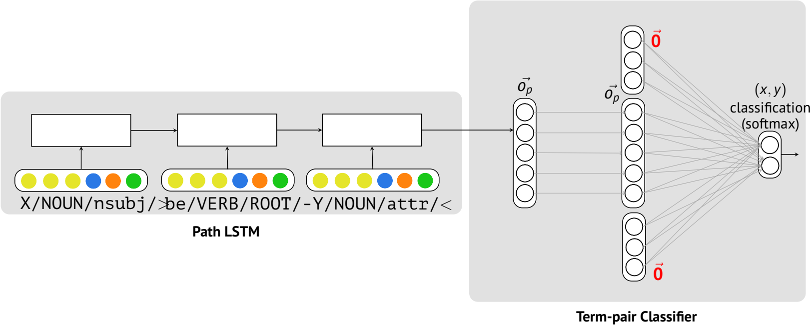
- The integrated method substantially outperforms both path-based and distributional methods

Analysis - Path Representation (1/2)

- Identify hypernymy-indicating paths:
  - Baselines: according to logistic regression feature weights

Analysis - Path Representation (1/2)

- Identify hypernymy-indicating paths:
  - Baselines: according to logistic regression feature weights
  - HypeNET: measure path contribution to positive classification:



- Take the top scoring paths according to $softmax(W \cdot [\vec{0}, \vec{o_p}, \vec{0}])[1]$

Analysis - Path Representation (2/2)

- Snow's method finds certain common paths:

  X company is a Y
  X ltd is a Y

Analysis - Path Representation (2/2)

- Snow's method finds certain common paths:

  X company is a Y
  X ltd is a Y

- PATTY-style generalizations find very general, possibly noisy paths:

  X NOUN is a Y

Analysis - Path Representation (2/2)

- Snow's method finds certain common paths:

  X company is a Y
  X ltd is a Y

- PATTY-style generalizations find very general, possibly noisy paths:

  X NOUN is a Y

- HypeNET makes fine-grained generalizations:

  X association is a Y
  X co. is a Y
  X company is a Y
  X corporation is a Y
  X foundation is a Y
  X group is a Y
  ...

Other Semantic Relations

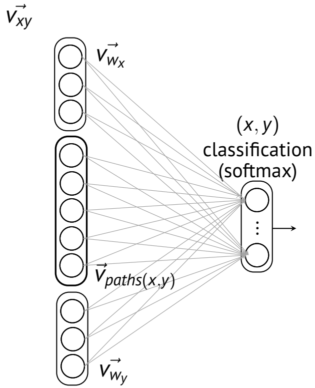
Recognizing Lexical Semantic Relations

- Given two terms, *x* and *y*, decide what is the semantic relation that holds between them (if any)
  - in some senses of *x* and *y*
  - e.g. both *fruit* and *company* are hypernyms of *apple*

LexNET - Multiple Semantic Relation Classification [Shwartz and Dagan, 2016a, Shwartz and Dagan, 2016b]

- Application of HypeNET for multiple relations:

  hypernymy, meroynymy, co-hyponymy, event, attribute, synonymy, antonymy, random



Results and Analysis

- LexNET outperforms individual path-based and distributional methods

Results and Analysis

- LexNET outperforms individual path-based and distributional methods
- Path-based contribution over distributional info is small when lexical memorization is enabled

Results and Analysis

- LexNET outperforms individual path-based and distributional methods
- Path-based contribution over distributional info is small when lexical memorization is enabled
- It is prominent in the following scenarios:
  - *x* or *y* are polysemous, e.g. *mero:(piano, key)*.

Results and Analysis

- LexNET outperforms individual path-based and distributional methods
- Path-based contribution over distributional info is small when lexical memorization is enabled
- It is prominent in the following scenarios:
  - *x* or *y* are polysemous, e.g. *mero:(piano, key)*.
  - the relation is not prototypical, e.g. *event:(cherry, pick)*.

Results and Analysis

- LexNET outperforms individual path-based and distributional methods
- Path-based contribution over distributional info is small when lexical memorization is enabled
- It is prominent in the following scenarios:
  - *x* or *y* are polysemous, e.g. *mero:(piano, key)*.
  - the relation is not prototypical, e.g. *event:(cherry, pick)*.
  - *x* or *y* are rare, e.g. *hyper:(mastodon, proboscidean)*.

Results and Analysis

- LexNET outperforms individual path-based and distributional methods
- Path-based contribution over distributional info is small when lexical memorization is enabled
- It is prominent in the following scenarios:
  - *x* or *y* are polysemous, e.g. *mero:(piano, key)*.
  - the relation is not prototypical, e.g. *event:(cherry, pick)*.
  - *x* or *y* are rare, e.g. *hyper:(mastodon, proboscidean)*.
- Thanks to the path representation, such relations are captured even with a single meaningful co-occurrence of *x* and *y*

Limitations

- All methods and baselines are bad in recognizing synonyms and antonyms.

Limitations

- All methods and baselines are bad in recognizing synonyms and antonyms.
    - **Path-based:**
        - Synonyms do not tend to occur together

Limitations

- All methods and baselines are bad in recognizing synonyms and antonyms.
    - **Path-based:**
        - Synonyms do not tend to occur together
        - Antonyms occur in similar paths as co-hyponyms:
          hot ***and*** *cold*, *cats* ***and*** *dogs*

Limitations

- All methods and baselines are bad in recognizing synonyms and antonyms.
  - **Path-based:**
    - Synonyms do not tend to occur together
    - Antonyms occur in similar paths as co-hyponyms:
      *hot **and** cold, cats **and** dogs*
  - **Distributional:**
    - Synonyms and antonyms occur in similar contexts:
      "go down in the elevator/lift", "it is hot/cold today"

Limitations

- All methods and baselines are bad in recognizing synonyms and antonyms.
  - **Path-based:**
    - Synonyms do not tend to occur together
    - Antonyms occur in similar paths as co-hyponyms: *hot* ***and*** *cold*, *cats* ***and*** *dogs*
  - **Distributional:**
    - Synonyms and antonyms occur in similar contexts: *"go down in the elevator/lift"*, *"it is hot/cold today"*

- [Nguyen et al., 2017] used the method successfully to distinguish only between synonyms and antonyms.

Learning Antonyms

- [Rajana et al., 2017] integrated morphological cues to distinguish antonymy from other relations:
  - Added a “negated” feature

Learning Antonyms

- [Rajana et al., 2017] integrated morphological cues to distinguish antonymy from other relations:
  - Added a “negated” feature
  - List of negated prefixes: *de, un, in, anti, il, non, dis*

Learning Antonyms

- [Rajana et al., 2017] integrated morphological cues to distinguish antonymy from other relations:
  - Added a “negated” feature
  - List of negated prefixes: *de, un, in, anti, il, non, dis*
  - A word with a negated prefix is replaced with the non-negated form, and the “negated” feature is turned on

Learning Antonyms

- [Rajana et al., 2017] integrated morphological cues to distinguish antonymy from other relations:
  - Added a “negated” feature
  - List of negated prefixes: *de, un, in, anti, il, non, dis*
  - A word with a negated prefix is replaced with the non-negated form, and the “negated” feature is turned on
  - e.g. *unhappy* $\rightarrow$ neg + *happy*

Learning Antonyms

- [Rajana et al., 2017] integrated morphological cues to distinguish antonymy from other relations:
  - Added a “negated” feature
  - List of negated prefixes: *de, un, in, anti, il, non, dis*
  - A word with a negated prefix is replaced with the non-negated form, and the “negated” feature is turned on
  - e.g. *unhappy* $\rightarrow$ neg + *happy*
- Improved performance on both binary and multiclass tasks

Interpreting Noun-Compounds

Interpreting Noun-Compounds: Task Definition

- Given a noun-compound $w_1 w_2$, classify the relation between the head w_2 and the modifier w_1
  - to one of a set of pre-defined relations

source

ground attack

olive oil

genetic defect

part of

sea bass

boat whistle

rotor head

purpose

baby oil

game room

service door

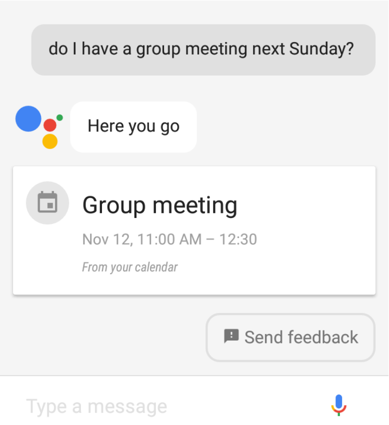
non-compositional

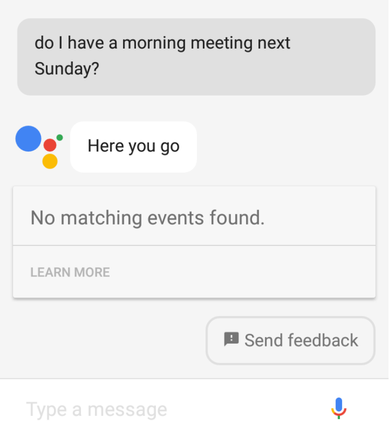
horse radish

baby sitting

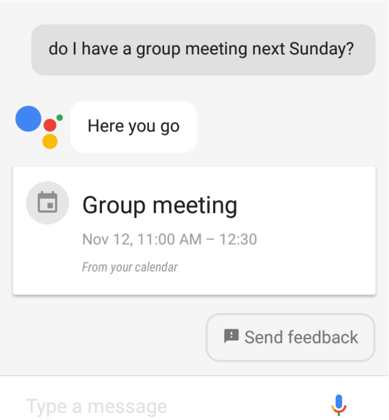
hot dog

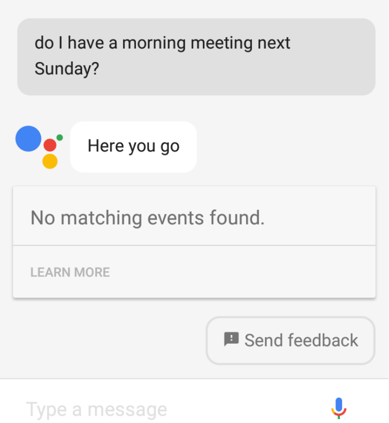
Motivation


do I have a group meeting next Sunday?
Here you go
Group meeting
Nov 12, 11:00 AM – 12:30
From your calendar
Send feedback
Type a message


do I have a morning meeting next Sunday?
Here you go
No matching events found.
LEARN MORE
Send feedback
Type a message

Motivation





- Need to interpret: *morning meeting* = meeting in the morning, *group meeting* = meeting with the group

Existing Methods

Distributional Approach

- Compute a vector for $w_1 w_2$ as a function of w_1 and w_2's vectors
  - $vec(olive\ oil) = f(vec(olive), vec(oil))$

Existing Methods

Distributional Approach

- Compute a vector for $w_1 w_2$ as a function of w_1 and w_2's vectors
  - $vec(olive\ oil) = f(vec(olive), vec(oil))$
  - Many ways to learn f [Mitchell and Lapata, 2010, Zanzotto et al., 2010, Dinu et al., 2013, Socher et al., 2012]

Existing Methods

Distributional Approach

- Compute a vector for $w_1 w_2$ as a function of w_1 and w_2's vectors
  - $vec(olive\ oil) = f(vec(olive), vec(oil))$
  - Many ways to learn f [Mitchell and Lapata, 2010, Zanzotto et al., 2010, Dinu et al., 2013, Socher et al., 2012]
- Use the noun-compound representation as a feature vector for classification [Dima, 2016]

Existing Methods
Distributional Approach

- Compute a vector for $w_1 w_2$ as a function of w_1 and w_2's vectors
  - $vec(olive\ oil) = f(vec(olive), vec(oil))$
  - Many ways to learn f [Mitchell and Lapata, 2010, Zanzotto et al., 2010, Dinu et al., 2013, Socher et al., 2012]

- Use the noun-compound representation as a feature vector for classification [Dima, 2016]
  - Best performance is achieved when $f(w_1, w_2) = [w_1; w_2]$

Existing Methods
Distributional Approach

- Compute a vector for $w_1 w_2$ as a function of w_1 and w_2's vectors
  - $vec(olive\ oil) = f(vec(olive), vec(oil))$
  - Many ways to learn f [Mitchell and Lapata, 2010, Zanzotto et al., 2010, Dinu et al., 2013, Socher et al., 2012]

- Use the noun-compound representation as a feature vector for classification [Dima, 2016]
  - Best performance is achieved when $f(w_1, w_2) = [w_1; w_2]$ 🤔
  - [Dima, 2016]: There is a **lexical memorization** issue

Existing Methods
Paraphrasing Approach

- [Nakov and Hearst, 2006]: the semantics of a noun-compound can be expressed with multiple paraphrases
  - *student protest* is a *protest* led by, be sponsored by, or be organized by *students*

Existing Methods
Paraphrasing Approach

- [Nakov and Hearst, 2006]: the semantics of a noun-compound can be expressed with multiple paraphrases
  - *student protest* is a *protest* led by, be sponsored by, or be organized by *students*

- Classification: use textual patterns / dependency paths of joint corpus occurrences of w_1 and w_2 as features
  - e.g.: [w_2] *obtained from* [w_1] *(oil obtained from olives)*

Existing Methods
Paraphrasing Approach

- [Nakov and Hearst, 2006]: the semantics of a noun-compound can be expressed with multiple paraphrases
  - *student protest* is a *protest* led by, be sponsored by, or be organized by *students*

- Classification: use textual patterns / dependency paths of joint corpus occurrences of w_1 and w_2 as features
  - e.g.: [w_2] *obtained from* [w_1] *(oil obtained from olives)*
  - There is a sparseness issue, e.g. [w_2] *extracted from* [w_1]

Interpreting Noun-Compounds
[Shwartz and Waterson, under review]

- We applied LexNET to this task

Interpreting Noun-Compounds
[Shwartz and Waterson, under review]

- We applied LexNET to this task
- LexNET improves performance:
  - On a lexical split dataset (i.e. not allowing lexical memorization)
  - On a new, challenging dataset we created

Interpreting Noun-Compounds
[Shwartz and Waterson, under review]

- We applied LexNET to this task
- LexNET improves performance:
  - On a lexical split dataset (i.e. not allowing lexical memorization)
  - On a new, challenging dataset we created
- Performs worse than the baseline when lexical memorization is possible

Interpreting Noun-Compounds
[Shwartz and Waterson, under review]

- We applied LexNET to this task
- LexNET improves performance:
  - On a lexical split dataset (i.e. not allowing lexical memorization)
  - On a new, challenging dataset we created
- Performs worse than the baseline when lexical memorization is possible
- In general, the task is very difficult:
  - Lots of relations
  - Some relations have no indicative paths (e.g. `non-compositional`)

Acquiring Predicate Paraphrases

Motivation

- Identify that various predicate mentions refer to the same event

Motivation

- Identify that various predicate mentions refer to the same event

- E.g. for recognizing textual entailment:
  - Text:
    Florida declares state of emergency evacuations as Irma intensifies to a category 5 storm

  - Hypothesis:
    Hurricane Irma strengthens to category 5 storm

Acquiring Predicate Paraphrases from News Tweets [Shwartz et al., 2017][2]

$[a]_0$ introduce $[a]_1$	$[a]_0$ welcome $[a]_1$
$[a]_0$ appoint $[a]_1$	$[a]_0$ to become $[a]_1$
$[a]_0$ die at $[a]_1$	$[a]_0$ pass away at $[a]_1$
$[a]_0$ hit $[a]_1$	$[a]_0$ sink to $[a]_1$
$[a]_0$ be investigate $[a]_1$	$[a]_0$ be probe $[a]_1$
$[a]_0$ eliminate $[a]_1$	$[a]_0$ slash $[a]_1$
$[a]_0$ announce $[a]_1$	$[a]_0$ unveil $[a]_1$
$[a]_0$ quit after $[a]_1$	$[a]_0$ resign after $[a]_1$
$[a]_0$ announce as $[a]_1$	$[a]_0$ to become $[a]_1$
$[a]_0$ threaten $[a]_1$	$[a]_0$ warn $[a]_1$
$[a]_0$ die at $[a]_1$	$[a]_0$ live until $[a]_1$
$[a]_0$ double down on $[a]_1$	$[a]_0$ stand by $[a]_1$
$[a]_0$ kill $[a]_1$	$[a]_0$ shoot $[a]_1$
$[a]_0$ approve $[a]_1$	$[a]_0$ pass $[a]_1$
seize $[a]_0$ at $[a]_1$	to grab $[a]_0$ at $[a]_1$

- Binary verbal predicate paraphrases

[2] Available at `https://github.com/vered1986/Chirps`

Acquiring Predicate Paraphrases from News Tweets [Shwartz et al., 2017][2]

$[a]_0$ introduce $[a]_1$	$[a]_0$ welcome $[a]_1$
$[a]_0$ appoint $[a]_1$	$[a]_0$ to become $[a]_1$
$[a]_0$ die at $[a]_1$	$[a]_0$ pass away at $[a]_1$
$[a]_0$ hit $[a]_1$	$[a]_0$ sink to $[a]_1$
$[a]_0$ be investigate $[a]_1$	$[a]_0$ be probe $[a]_1$
$[a]_0$ eliminate $[a]_1$	$[a]_0$ slash $[a]_1$
$[a]_0$ announce $[a]_1$	$[a]_0$ unveil $[a]_1$
$[a]_0$ quit after $[a]_1$	$[a]_0$ resign after $[a]_1$
$[a]_0$ announce as $[a]_1$	$[a]_0$ to become $[a]_1$
$[a]_0$ threaten $[a]_1$	$[a]_0$ warn $[a]_1$
$[a]_0$ die at $[a]_1$	$[a]_0$ live until $[a]_1$
$[a]_0$ double down on $[a]_1$	$[a]_0$ stand by $[a]_1$
$[a]_0$ kill $[a]_1$	$[a]_0$ shoot $[a]_1$
$[a]_0$ approve $[a]_1$	$[a]_0$ pass $[a]_1$
seize $[a]_0$ at $[a]_1$	to grab $[a]_0$ at $[a]_1$

- Binary verbal predicate paraphrases
- Extracted from Twitter

[2] Available at `https://github.com/vered1986/Chirps`

Acquiring Predicate Paraphrases from News Tweets [Shwartz et al., 2017][2]

$[a]_0$ introduce $[a]_1$	$[a]_0$ welcome $[a]_1$
$[a]_0$ appoint $[a]_1$	$[a]_0$ to become $[a]_1$
$[a]_0$ die at $[a]_1$	$[a]_0$ pass away at $[a]_1$
$[a]_0$ hit $[a]_1$	$[a]_0$ sink to $[a]_1$
$[a]_0$ be investigate $[a]_1$	$[a]_0$ be probe $[a]_1$
$[a]_0$ eliminate $[a]_1$	$[a]_0$ slash $[a]_1$
$[a]_0$ announce $[a]_1$	$[a]_0$ unveil $[a]_1$
$[a]_0$ quit after $[a]_1$	$[a]_0$ resign after $[a]_1$
$[a]_0$ announce as $[a]_1$	$[a]_0$ to become $[a]_1$
$[a]_0$ threaten $[a]_1$	$[a]_0$ warn $[a]_1$
$[a]_0$ die at $[a]_1$	$[a]_0$ live until $[a]_1$
$[a]_0$ double down on $[a]_1$	$[a]_0$ stand by $[a]_1$
$[a]_0$ kill $[a]_1$	$[a]_0$ shoot $[a]_1$
$[a]_0$ approve $[a]_1$	$[a]_0$ pass $[a]_1$
seize $[a]_0$ at $[a]_1$	to grab $[a]_0$ at $[a]_1$

- Binary verbal predicate paraphrases
- Extracted from Twitter
- Ever-growing resource: currently around 1.5M paraphrases

[2] Available at `https://github.com/vered1986/Chirps`

Assumptions

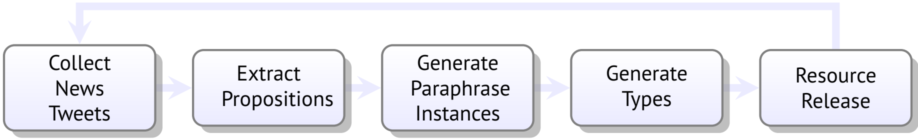
- **Main assumption:** redundant news headlines of the same event are likely to describe it with different words [Shinyama et al., 2002, Barzilay and Lee, 2003].

Assumptions

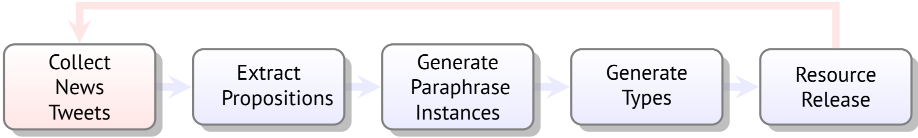
- **Main assumption:** redundant news headlines of the same event are likely to describe it with different words [Shinyama et al., 2002, Barzilay and Lee, 2003].
- **This work:** propositions extracted from tweets discussing news events, published on the same day, that agree on their arguments, are predicate paraphrases.

[Amazon] to buy / is buying / to acquire [Whole Foods]

Resource Collection



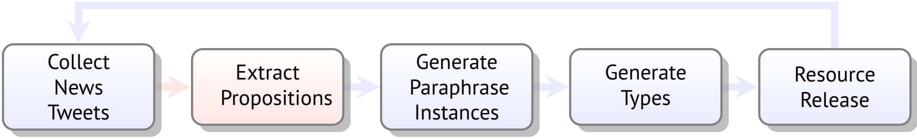
Resource Collection



- Query the Twitter Search API for news tweets in English

Amazon is buying Whole Foods in $13.7B
Amazon to acquire Whole Foods Market in deal valued at nearly $14 billion
...

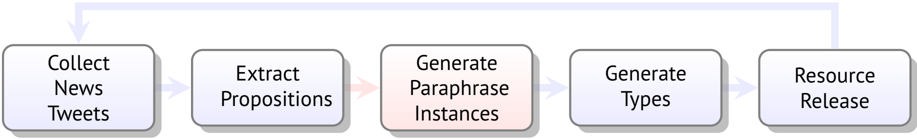
Resource Collection



- Extract propositions from tweets using PropS [Stanovsky et al., 2016]
- Get binary verbal predicate templates, and apply argument reduction [Stanovsky and Dagan, 2016]

[Amazon] **buy** [Whole Foods]
[Amazon] **acquire** [Whole Foods Market]
. . .

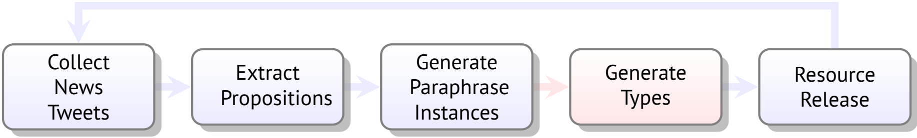
Resource Collection



- We consider two predicates as paraphrases if:
  1. They appear on the same day.
  2. Each of their arguments aligns with a unique argument in the other predicate.
- Two levels of argument matching: **strict** (exact match / short edit distance) and **loose** (partial token matching / WordNet synonyms)

$[a]_0$ **buy** $[a]_1$	$[a]_0$ **acquire** $[a]_1$	Amazon	Whole Foods
$[a]_0$ **buy** $[a]_1$	$[a]_0$ **acquire** $[a]_1$	Intel	Mobileye
	...		

Resource Collection



Heuristic score for a predicate paraphrase type:

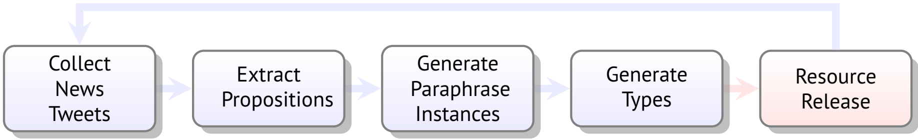
$$p_1 = [a]_0 \textbf{ buy } [a]_1, \quad p_2 = [a]_0 \textbf{ acquire } [a]_1$$

$$s(p_1, p_2) = count(p_1, p_2) \cdot \left(1 + \frac{days(p_1, p_2)}{N}\right)$$

- $count(p_1, p_2)$ assigns high scores for frequent paraphrases
- N - number of days since the resource collection begun
- $\frac{days(p_1, p_2)}{N}$ eliminates noise from two arguments participating in different events on the same day

  1) *Last year when Chuck Berry turned 90*; 2) *Chuck Berry dies at 90*

Resource Collection



- We release our resource daily, with two files:
  - **Instances**: predicates, arguments and tweet IDs.
  - **Types**: predicate paraphrase pair types ranked in a descending order according to the heuristic accuracy score.

Thank You!

References I

[Baroni et al., 2012] Baroni, M., Bernardi, R., Do, N.-Q., and Shan, C.-c. (2012). Entailment above the word level in distributional semantics. In *EACL*, pages 23–32.

[Barzilay and Lee, 2003] Barzilay, R. and Lee, L. (2003). Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *NAACL*.

[Dima, 2016] Dima, C. (2016). *Proceedings of the 1st Workshop on Representation Learning for NLP*, chapter On the Compositionality and Semantic Interpretation of English Noun Compounds, pages 27–39. Association for Computational Linguistics.

[Dinu et al., 2013] Dinu, G., Pham, N. T., and Baroni, M. (2013). General estimation and evaluation of compositional distributional semantic models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 50–58, Sofia, Bulgaria. Association for Computational Linguistics.

[Harris, 1954] Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

[Hearst, 1992] Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *ACL*, pages 539–545.

[Levy et al., 2015] Levy, O., Remus, S., Biemann, C., and Dagan, I. (2015). Do supervised distributional methods really learn lexical inference relations. *NAACL*.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

[Mitchell and Lapata, 2010] Mitchell, J. and Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.

[Nakashole et al., 2012] Nakashole, N., Weikum, G., and Suchanek, F. (2012). Patty: a taxonomy of relational patterns with semantic types. In *EMNLP and CoNLL*, pages 1135–1145.

[Nakov and Hearst, 2006] Nakov, P. and Hearst, M. (2006). Using verbs to characterize noun-noun relations. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 233–244. Springer.

References II

[Nguyen et al., 2017] Nguyen, K. A., Schulte im Walde, S., and Vu, N. T. (2017). Distinguishing antonyms and synonyms in a pattern-based neural network. In *EACL*.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

[Rajana et al., 2017] Rajana, S., Callison-Burch, C., Apidianaki, M., and Shwartz, V. (2017). Learning antonyms with paraphrases and a morphology-aware neural network. In **SEM*, pages 12–21.

[Roller et al., 2014] Roller, S., Erk, K., and Boleda, G. (2014). Inclusive yet selective: Supervised distributional hypernymy detection. In *COLING*, pages 1025–1036.

[Shinyama et al., 2002] Shinyama, Y., Sekine, S., and Sudo, K. (2002). Automatic paraphrase acquisition from news articles. In *HLT*, pages 313–318. Morgan Kaufmann Publishers Inc.

[Shwartz and Dagan, 2016a] Shwartz, V. and Dagan, I. (2016a). path-based vs. distributional information in recognizing lexical semantic relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex-V), in COLING*, Osaka, Japan.

[Shwartz and Dagan, 2016b] Shwartz, V. and Dagan, I. (2016b). cogalex-v shared task: Lexnet - integrated path-based and distributional method for the identification of semantic relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex-V), in COLING*, Osaka, Japan.

[Shwartz et al., 2016] Shwartz, V., Goldberg, Y., and Dagan, I. (2016). Improving hypernymy detection with an integrated path-based and distributional method. In *ACL*, pages 2389–2398.

[Shwartz et al., 2017] Shwartz, V., Stanovsky, G., and Dagan, I. (2017). Acquiring predicate paraphrases from news tweets. In **SEM*, pages 155–160.

[Snow et al., 2004] Snow, R., Jurafsky, D., and Ng, A. Y. (2004). Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

References III

[Socher et al., 2012] Socher, R., Huval, B., Manning, C. D., and Ng, A. Y. (2012). Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 1201–1211. Association for Computational Linguistics.

[Stanovsky and Dagan, 2016] Stanovsky, G. and Dagan, I. (2016). Annotating and predicting non-restrictive noun phrase modifications. In *ACL*.

[Stanovsky et al., 2016] Stanovsky, G., Ficler, J., Dagan, I., and Goldberg, Y. (2016). Getting more out of syntax with props. *arXiv*.

[Weeds et al., 2014] Weeds, J., Clarke, D., Reffin, J., Weir, D., and Keller, B. (2014). Learning to distinguish hypernyms and co-hyponyms. In *COLING*, pages 2249–2259.

[Zanzotto et al., 2010] Zanzotto, F. M., Korkontzelos, I., Fallucchi, F., and Manandhar, S. (2010). Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1263–1271. Association for Computational Linguistics.