Making Sense of English Nominal Compounds

Murhaf Fares
Language Technology Group
Department of Informatics
University of Oslo

February 14, 2017

Nominal Compounds

- [Li, 1971]: “the simple concatenation of any two or more nouns functioning as a third nominal”

They simply got carried away with interpreting what [$_{ARG0}$ the *executive order*] [$_{rel}$ banning] [$_{ARG1}$ assassinations] really meant. (ProbBank)

Nominal Compounds

- [Li, 1971]: “the simple concatenation of any two or more nouns functioning as a third nominal”

They simply got carried away with interpreting what [$_{ARG0}$ the *executive order*] [$_{rel}$ banning] [$_{ARG1}$ assassinations] really meant. (ProbBank)

Motivation

- Frequent:
  - 3% of all words in the British National Corpus [Ó Séaghdha, 2008]
  - 3.9% of all words in Reuters corpus [Baldwin and Tanaka, 2004]
- Productive: *executive order*, *purchase order*, *delivery order*
- . . . but also: human door, people door

Motivation

- Frequent:
  - 3% of all words in the British National Corpus [Ó Séaghdha, 2008]
  - 3.9% of all words in Reuters corpus [Baldwin and Tanaka, 2004]
- Productive: *executive order*, *purchase order*, *delivery order*
- . . . but also: human door, people door

Motivation

- Frequent:
  - 3% of all words in the British National Corpus [Ó Séaghdha, 2008]
  - 3.9% of all words in Reuters corpus [Baldwin and Tanaka, 2004]
- Productive: *executive order*, *purchase order*, *delivery order*
- . . . but also: human door, people door

Motivation

- Frequent:
  - 3% of all words in the British National Corpus [Ó Séaghdha, 2008]
  - 3.9% of all words in Reuters corpus [Baldwin and Tanaka, 2004]
- Productive: *executive order*, *purchase order*, *delivery order*
- . . . but also: human door, people door

FIGURE 3.

Figure: [Downing, 1977]

Background: Three Tasks

Three NLP tasks related to noun compounds [Lauer and Dras, 1994]

- Detection or identification of noun compounds
- Syntactic analysis of the internal structure, i.e. left vs. right bracketing of compounds with more than two constituents
- *Interpretation of the semantic relation holding between the constituents of the compound*

Background: Three Tasks

Three NLP tasks related to noun compounds [Lauer and Dras, 1994]

- Detection or identification of noun compounds
- Syntactic analysis of the internal structure, i.e. left vs. right bracketing of compounds with more than two constituents
- *Interpretation of the semantic relation holding between the constituents of the compound*

Background: Three Tasks

Three NLP tasks related to noun compounds [Lauer and Dras, 1994]

- Detection or identification of noun compounds
- Syntactic analysis of the internal structure, i.e. left vs. right bracketing of compounds with more than two constituents
- *Interpretation of the semantic relation holding between the constituents of the compound*

Two Main Approaches

Two main approaches to semantic interpretation of nominal compounds:

- Taxonomy-based [Girju et al., 2005, Tratz and Hovy, 2010, Ó Séaghdha and Copestake, 2013]
- Paraphrase-based [Nakov, 2013]

Two Main Approaches

Two main approaches to semantic interpretation of nominal compounds:

- Taxonomy-based [Girju et al., 2005, Tratz and Hovy, 2010, Ó Séaghdha and Copestake, 2013]
- Paraphrase-based [Nakov, 2013]

Background: Datasets

Dataset	Size	Relations
Nastase & Szpakowicz (2003)	600	30
Girju et al. (2005)	4,500	21
Ó Séaghdha & Copestake (2007)	1,443	6
Kim & Baldwin (2008)	2,169	20
Tratz & Hovy (2010)	17,509	43
Nombank (Fares 2016)	10,596	20
Functor (Fares 2016)	10,596	35

Table: Overview of noun compound datasets. Size: type count

Background: Datasets – Examples

- *cancer death*
  - Tratz: `CREATOR-PROVIDER-CAUSE_OF`
  - Ó Séaghdha: `INST`
  - Nombank: `ARGM-CAU`
  - Functor: `CAUS`
- *world opinion*
  - Tratz: `EXPERIENCER-OF-EXPERIENCE`
  - Ó Séaghdha: `HAVE`
  - Nombank: `ARG0`
  - Functor: `ACT-arg`

Background: Datasets – Examples

- *cancer death*
  - Tratz: `CREATOR-PROVIDER-CAUSE_OF`
  - Ó Séaghdha: `INST`
  - Nombank: `ARGM-CAU`
  - Functor: `CAUS`
- *world opinion*
  - Tratz: `EXPERIENCER-OF-EXPERIENCE`
  - Ó Séaghdha: `HAVE`
  - Nombank: `ARG0`
  - Functor: `ACT-arg`

Background: Datasets – Examples

- *research team*
  - Tratz: `PERFORM&ENGAGE_IN`
  - Ó Séaghdha: `ACTOR`
  - Nombank: `ARG1`
  - Functor: `RSTR`
- *aid package*
  - Tratz: `TOPIC`
  - Ó Séaghdha: `INST`
  - Nombank: `ARG1`
  - Functor: `RSTR`

Background: Datasets – Examples

- *research team*
  - Tratz: `PERFORM&ENGAGE_IN`
  - Ó Séaghdha: `ACTOR`
  - Nombank: `ARG1`
  - Functor: `RSTR`
- *aid package*
  - Tratz: `TOPIC`
  - Ó Séaghdha: `INST`
  - Nombank: `ARG1`
  - Functor: `RSTR`

Background: Classification Approaches

- Maximum Entropy [Tratz and Hovy, 2010]
- Support Vector Machines [Ó Séaghdha and Copestake, 2009]
- Deep Neural Networks [Dima and Hinrichs, 2015]
  - Used word vectors from "a selection of publicly available word embeddings" as input to a neural network.

Background: Classification Approaches

- Maximum Entropy [Tratz and Hovy, 2010]
- Support Vector Machines [Ó Séaghdha and Copestake, 2009]
- Deep Neural Networks [Dima and Hinrichs, 2015]
  - Used word vectors from “a selection of publicly available word embeddings” as input to a neural network.

Background: Classification Approaches

- Maximum Entropy [Tratz and Hovy, 2010]
- Support Vector Machines [Ó Séaghdha and Copestake, 2009]
- Deep Neural Networks [Dima and Hinrichs, 2015]
  - Used word vectors from "a selection of publicly available word embeddings" as input to a neural network.

Background: Classification Approaches

- Maximum Entropy [Tratz and Hovy, 2010]
- Support Vector Machines [Ó Séaghdha and Copestake, 2009]
- Deep Neural Networks [Dima and Hinrichs, 2015]
  - Used word vectors from “a selection of publicly available word embeddings” as input to a neural network.

The Big Questions

Do *word embeddings* capture the semantic relations holding between the constituents of nominal compounds?

Can we predict the compound semantic relations using the *vector arithmetic* typically used to solve word analogy tasks?

The Big Questions

Do *word embeddings* capture the semantic relations holding between the constituents of nominal compounds?

Can we predict the compound semantic relations using the *vector arithmetic* typically used to solve word analogy tasks?

Vector Space Models

- Vector space representations of words (meaning) based on the distributional hypothesis
- Words are represented as vectors of real numbers in $\mathbb{R}^d$
  - Corresponding to number of times w_j occur in the *context* of w_i
  - The vectors are referred to as the *co-occurrence matrix*
- Similarity measures: Euclidean distance, cosine similarity, etc.
- Typically very high-dimensional sparse models

Vector Space Models

- Vector space representations of words (meaning) based on the distributional hypothesis
- Words are represented as vectors of real numbers in $\mathbb{R}^d$
  - Corresponding to number of times w_j occur in the *context* of w_i
  - The vectors are referred to as the *co-occurrence matrix*
- Similarity measures: Euclidean distance, cosine similarity, etc.
- Typically very high-dimensional sparse models

Vector Space Models

- Vector space representations of words (meaning) based on the distributional hypothesis
- Words are represented as vectors of real numbers in $\mathbb{R}^d$
  - Corresponding to number of times w_j occur in the *context* of w_i
  - The vectors are referred to as the *co-occurrence matrix*
- Similarity measures: Euclidean distance, cosine similarity, etc.
- Typically very high-dimensional sparse models

Vector Space Models

- Vector space representations of words (meaning) based on the distributional hypothesis
- Words are represented as vectors of real numbers in $\mathbb{R}^d$
  - Corresponding to number of times w_j occur in the *context* of w_i
  - The vectors are referred to as the *co-occurrence matrix*
- Similarity measures: Euclidean distance, cosine similarity, etc.
- Typically very high-dimensional sparse models

Vector Space Models

- Vector space representations of words (meaning) based on the distributional hypothesis
- Words are represented as vectors of real numbers in $\mathbb{R}^d$
  - Corresponding to number of times w_j occur in the *context* of w_i
  - The vectors are referred to as the *co-occurrence matrix*
- Similarity measures: Euclidean distance, cosine similarity, etc.
- Typically very high-dimensional sparse models

Word Embeddings

- Word embeddings are vector space models with:
  - Lower-dimensional dense vectors
- Many approaches and tools: CBOW, SG (word2vec), *GloVe*

Word Embeddings

- Word embeddings are vector space models with:
  - Lower-dimensional dense vectors
- Many approaches and tools: CBOW, SG (word2vec), *GloVe*

Word Embeddings

- Word embeddings are vector space models with:
  - Lower-dimensional dense vectors
- Many approaches and tools: CBOW, SG (word2vec), *GloVe*

GloVe: Global Vectors for Word Representation

- Let X be a word-word co-occurrence matrix
- X_{ij}: the number of times word j occurs in the context of word i
- $P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}$

GloVe: Global Vectors for Word Representation

- Let X be a word-word co-occurrence matrix
- X_{ij}: the number of times word j occurs in the context of word i
- $P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}$

GloVe: Global Vectors for Word Representation

- Let X be a word-word co-occurrence matrix
- X_{ij}: the number of times word j occurs in the context of word i
- $P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}$

GloVe: Ratios

GloVe relies on ratio of co-occurrence probabilities instead of just co-occurrence probabilities

Prob. & ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Table: Based on Table 1 in [Pennington et al., 2014]

GloVe: Ratios

GloVe relies on ratio of co-occurrence probabilities instead of just co-occurrence probabilities

Prob. & ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Table: Based on Table 1 in [Pennington et al., 2014]

GloVe: Vector Learning

Given the observation about ratio of co-occurrence probabilities

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

. . . fast forward seven steps

$$J = \sum_{i,j=1}^{V} f(X_{ij})\left(w_j^T \tilde{w}_j + b_i + \tilde{b}_j - log X_{ij}\right)^2$$

See [Pennington et al., 2014]

GloVe: Vector Learning

Given the observation about ratio of co-occurrence probabilities

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

. . . fast forward seven steps

$$J = \sum_{i,j=1}^{V} f(X_{ij})\left(w_j^T \tilde{w}_j + b_i + \tilde{b}_j - logX_{ij}\right)^2$$

See [Pennington et al., 2014]

GloVe: Training

Given a training corpus and a set of parameters (to be discussed later):

1. Construct a vocabulary dictionary
2. Construct a co-occurrence matrix
3. Shuffle the co-occurrence matrix
4. Train the GloVe model (using the equation from the previous slide)

GloVe: Training

Given a training corpus and a set of parameters (to be discussed later):

1. Construct a vocabulary dictionary
2. Construct a co-occurrence matrix
3. Shuffle the co-occurrence matrix
4. Train the GloVe model (using the equation from the previous slide)

GloVe: Training

Given a training corpus and a set of parameters (to be discussed later):

1. Construct a vocabulary dictionary
2. Construct a co-occurrence matrix
3. Shuffle the co-occurrence matrix
4. Train the GloVe model (using the equation from the previous slide)

GloVe: Training

Given a training corpus and a set of parameters (to be discussed later):

1. Construct a vocabulary dictionary
2. Construct a co-occurrence matrix
3. Shuffle the co-occurrence matrix
4. Train the GloVe model (using the equation from the previous slide)

GloVe: Parameters and Hyperparameters

- Text pre-processing:
  - sentence segmentation
  - tokenization
  - lemmatization
- Vocabulary:
  - frequency cutoff
- Co-occurrence matrix:
  - context window size
  - (a)symmetric window
- Training:
  - vector dimensions
  - number of iterations
  - learning rate
  - ...

GloVe: Parameters and Hyperparameters

- Text pre-processing:
  - sentence segmentation
  - tokenization
  - lemmatization
- Vocabulary:
  - frequency cutoff
- Co-occurrence matrix:
  - context window size
  - (a)symmetric window
- Training:
  - vector dimensions
  - number of iterations
  - learning rate
  - . . .

GloVe: Parameters and Hyperparameters

- Text pre-processing:
  - sentence segmentation
  - tokenization
  - lemmatization
- Vocabulary:
  - frequency cutoff
- Co-occurrence matrix:
  - context window size
  - (a)symmetric window
- Training:
  - vector dimensions
  - number of iterations
  - learning rate
  - ...

GloVe: Parameters and Hyperparameters

- Text pre-processing:
  - sentence segmentation
  - tokenization
  - lemmatization
- Vocabulary:
  - frequency cutoff
- Co-occurrence matrix:
  - context window size
  - (a)symmetric window
- Training:
  - vector dimensions
  - number of iterations
  - learning rate
  - ...

Vector Arithmetic

Can we predict the compound semantic relations using the vector arithmetic typically used to solve word analogy tasks?

King is to queen as man is to ?

$$\text{3COSADD:} \quad \arg\max_{b^* \in V}(cos(b^*, b - a + a^*))$$

$$\text{PAIRDIRECTION:} \quad \arg\max_{b^* \in V}(cos(b^* - b, a^* - a))$$

[Levy and Goldberg, 2014] report that [Mikolov et al., 2013] used 3COSADD to solve the syntactic analogies task and PAIRDIRECTION to solve the semantic one.

Vector Arithmetic

Can we predict the compound semantic relations using the vector arithmetic typically used to solve word analogy tasks?

King is to queen as man is to ?

$$\text{3CosAdd:} \quad \arg\max_{b^* \in V}(cos(b^*, b - a + a^*))$$

$$\text{PairDirection:} \quad \arg\max_{b^* \in V}(cos(b^* - b, a^* - a))$$

[Levy and Goldberg, 2014] report that [Mikolov et al., 2013] used 3CosAdd to solve the syntactic analogies task and PairDirection to solve the semantic one.

Vector Arithmetic

Can we predict the compound semantic relations using the vector arithmetic typically used to solve word analogy tasks?

King is to queen as man is to ?

$$\text{3CosAdd:} \quad \arg\max_{b^* \in V}(cos(b^*, b - a + a^*))$$

$$\text{PairDirection:} \quad \arg\max_{b^* \in V}(cos(b^* - b, a^* - a))$$

[Levy and Goldberg, 2014] report that [Mikolov et al., 2013] used 3CosAdd to solve the syntactic analogies task and PairDirection to solve the semantic one.

Vector Arithmetic

Can we predict the compound semantic relations using the vector arithmetic typically used to solve word analogy tasks?

King is to queen as man is to ?

$$\text{3COSADD:} \quad \arg\max_{b^* \in V}(cos(b^*, b - a + a^*))$$

$$\text{PAIRDIRECTION:} \quad \arg\max_{b^* \in V}(cos(b^* - b, a^* - a))$$

[Levy and Goldberg, 2014] report that [Mikolov et al., 2013] used 3COSADD to solve the syntactic analogies task and PAIRDIRECTION to solve the semantic one.

Vector Arithmetic

Can we predict the compound semantic relations using the vector arithmetic typically used to solve word analogy tasks?

King is to queen as man is to ?

$$\text{3CosAdd:} \quad \arg\max_{b^* \in V}(cos(b^*, b - a + a^*))$$

$$\text{PairDirection:} \quad \arg\max_{b^* \in V}(cos(b^* - b, a^* - a))$$

[Levy and Goldberg, 2014] report that [Mikolov et al., 2013] used 3CosAdd to solve the syntactic analogies task and PairDirection to solve the semantic one.

Vector Arithmetic for Nominal Compounds

Given a test compound b, b^*

$$\text{3CosAdd:} \quad \arg\max_{a,a^* \in C}(cos(b^*, b - a + a^*))$$

$$\text{PairDirection:} \quad \arg\max_{a,a^* \in C}(cos(b^* - b, a^* - a))$$

C is the set training compounds

Return the *relation* of the most similar compound

Vector Arithmetic for Nominal Compounds

Given a test compound b, b^*

$$\text{3CosAdd:}\quad \arg\max_{a,a^* \in C}(cos(b^*, b - a + a^*))$$

$$\text{PairDirection:}\quad \arg\max_{a,a^* \in C}(cos(b^* - b, a^* - a))$$

C is the set training compounds

Return the *relation* of the most similar compound

Vector Arithmetic for Nominal Compounds

Given a test compound b, b^*

$$\text{3COSADD:} \quad \underset{a,a^* \in C}{\arg\max}(cos(b^*, b - a + a^*))$$

$$\text{PAIRDIRECTION:} \quad \underset{a,a^* \in C}{\arg\max}(cos(b^* - b, a^* - a))$$

C is the set training compounds

Return the *relation* of the most similar compound

Experimental Setup: Datasets

Dataset	Size	Relations
Ó Séaghdha	1,443	6
Tratz (2011)	19,027	37
Nombank	7,850	17
Functors	7,850	33

Table: Overview of noun compound datasets.

- 5-fold cross validation: Ó Séaghdha
- 10-fold cross validation: Tratz, Nombank, Functor
- Numbers reported are accuracy, unless otherwise said

Experimental Setup: Datasets

Dataset	Size	Relations
Ó Séaghdha	1,443	6
Tratz (2011)	19,027	37
Nombank	7,850	17
Functors	7,850	33

Table: Overview of noun compound datasets.

- 5-fold cross validation: Ó Séaghdha
- 10-fold cross validation: Tratz, Nombank, Functor
- Numbers reported are accuracy, unless otherwise said

Experimental Setup: Tools

- Text pre-processing: Stanford CoreNLP
- Word embeddings: GloVe
- Vector arithmetic & evaluation: Python script(s)

Experimental Setup: Tools

- Text pre-processing: Stanford CoreNLP
- Word embeddings: GloVe
- Vector arithmetic & evaluation: Python script(s)

Experimental Setup: Tools

- Text pre-processing: Stanford CoreNLP
- Word embeddings: GloVe
- Vector arithmetic & evaluation: Python script(s)

Five Sets of Experiments

1. Pre-processing: lemma vs. full form
2. Training data: Wikipedia, Gigaword, both
3. Vector dimension: 50, 100, 300, 600, 1000
4. Vector arithmetic: 3CosAdd vs. PairDirection
5. 1-nearest neighbor vs. k-nearest neighbors

Five Sets of Experiments

1. Pre-processing: lemma vs. full form
2. Training data: Wikipedia, Gigaword, both
3. Vector dimension: 50, 100, 300, 600, 1000
4. Vector arithmetic: 3CosAdd vs. PairDirection
5. 1-nearest neighbor vs. k-nearest neighbors

Five Sets of Experiments

1. Pre-processing: lemma vs. full form
2. Training data: Wikipedia, Gigaword, both
3. Vector dimension: 50, 100, 300, 600, 1000
4. Vector arithmetic: 3COSADD vs. PAIRDIRECTION
5. 1-nearest neighbor vs. k-nearest neighbors

Five Sets of Experiments

1. Pre-processing: lemma vs. full form
2. Training data: Wikipedia, Gigaword, both
3. Vector dimension: 50, 100, 300, 600, 1000
4. Vector arithmetic: 3CosAdd vs. PairDirection
5. 1-nearest neighbor vs. k-nearest neighbors

Five Sets of Experiments

1. Pre-processing: lemma vs. full form
2. Training data: Wikipedia, Gigaword, both
3. Vector dimension: 50, 100, 300, 600, 1000
4. Vector arithmetic: 3CosAdd vs. PairDirection
5. 1-nearest neighbor vs. k-nearest neighbors

1. Text Pre-processing: Lemma vs. Full Form

	Tratz	Ó Séaghdha	Nombank	Functor
Full form	57.24	45.14	66.43	42.46
Lemma	56.08	45.21	67.03	42.54

Table: Lemma vs. full form. Wiki+Giga, $300d$, PAIRDIRECTION

More or less the same, but . . .

shorter training time for lemma-based models and . . .

1. Text Pre-processing: Lemma vs. Full Form

	Tratz	Ó Séaghdha	Nombank	Functor
Full form	57.24	45.14	66.43	42.46
Lemma	56.08	45.21	67.03	42.54

Table: Lemma vs. full form. Wiki+Giga, $300d$, PAIRDIRECTION

More or less the same, but . . .

shorter training time for lemma-based models and . . .

Aside: Lemma vs. Full Form

Model	SimLex	Analogy
GloVe wiki lemmas	0.36	0.81
GloVe wiki forms	0.31	0.81
GloVe giga lemmas	0.38	0.72
GloVe giga forms	0.32	0.71
GloVe comb lemmas	0.40	0.77
GloVe comb forms	0.35	0.76

Table: Benchmarking against SimLex-999 and the Google Analogies Dataset

2. Size of Training Data

	Corpus	Tratz	Ó Séaghdha	Nombank	Functor
PAIRDIRECTION	Wikipeida	55.99	**45.56**	65.36	41.57
	Gigaword	**56.34**	43.75	**67.17**	**43.29**
	Wiki+Giga	56.08	45.21	67.03	42.54
3COSADD	Wikipeida	54.57	**40.14**	64.05	41.45
	Gigaword	**55.25**	37.92	**67.49**	**44.09**
	Wiki+Giga	55.22	39.65	66.91	43.72

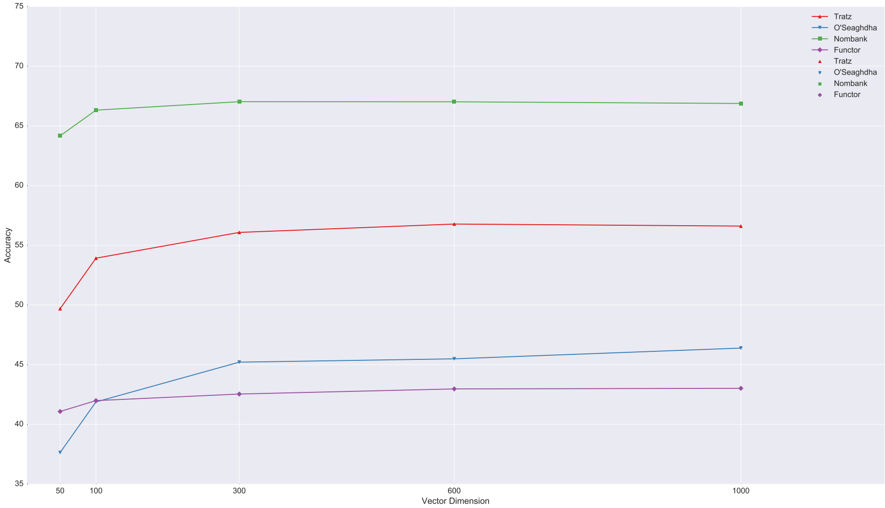
Table: The impact of the size of training data on accuracy. Other parameters: lemma, $300d$

Wikipedia: 1.08 billion tokens

Gigaword: 2.47 billion tokens

Wiki+Giga: 3.56 billion tokens

3. Vector Dimensionality



Other model parameters: lemma, Wiki+Giga, PAIRDIRECTION

4. Vector Arithmetics: PairDirection vs. 3CosAdd

Reminder:

$$\text{3CosAdd} \quad \arg\max_{b^* \in V}(cos(b^*, b - a + a^*))$$

$$\text{PairDirection} \quad \arg\max_{b^* \in V}(cos(b^* - b, a^* - a))$$

	Tratz	Ó Séaghdha	Nombank	Functor
PairDirection	**56.08**	**45.21**	**67.03**	42.54
3CosAdd	55.22	39.65	66.91	**43.72**

Other parameters: lemma, Wiki+Giga, $300d$

4. Vector Arithmetics: PairDirection vs. 3CosAdd

Reminder:

3CosAdd $\arg\max_{b^* \in V}(cos(b^*, b - a + a^*))$

PairDirection $\arg\max_{b^* \in V}(cos(b^* - b, a^* - a))$

	Tratz	Ó Séaghdha	Nombank	Functor
PairDirection	**56.08**	**45.21**	**67.03**	42.54
3CosAdd	55.22	39.65	66.91	**43.72**

Other parameters: lemma, Wiki+Giga, $300d$

5. 1-NN vs. k-NN

	Tratz	Ó Séaghdha	Nombank	Functor
$n = 1$	56.08	45.21	67.03	42.54
$n = 3$	58.58	46.39	71.03	45.71
$n = 5$	60.28	45.7	72.06	47.8
$n = 7$	60.84	44.24	72.38	49.8
$n = 9$	*61.12*	*45.69*	*72.21*	*51.08*
$n = 11$	61.35	44.72	72.18	51.7

Table: 1-NN vs. k-NN. Other parameters: lemma, $300d$, Wiki+Giga, PAIRDIRECTION

Clear increase in accuracy, but . . .

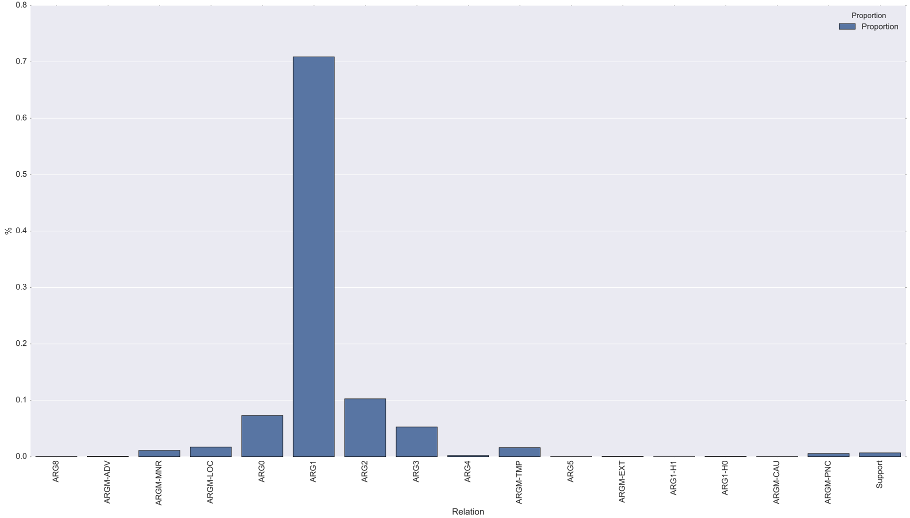
5. 1-NN vs. k-NN

	Tratz	Ó Séaghdha	Nombank	Functor
$n = 1$	56.08	45.21	67.03	42.54
$n = 3$	58.58	46.39	71.03	45.71
$n = 5$	60.28	45.7	72.06	47.8
$n = 7$	60.84	44.24	72.38	49.8
$n = 9$	*61.12*	*45.69*	*72.21*	*51.08*
$n = 11$	61.35	44.72	72.18	51.7

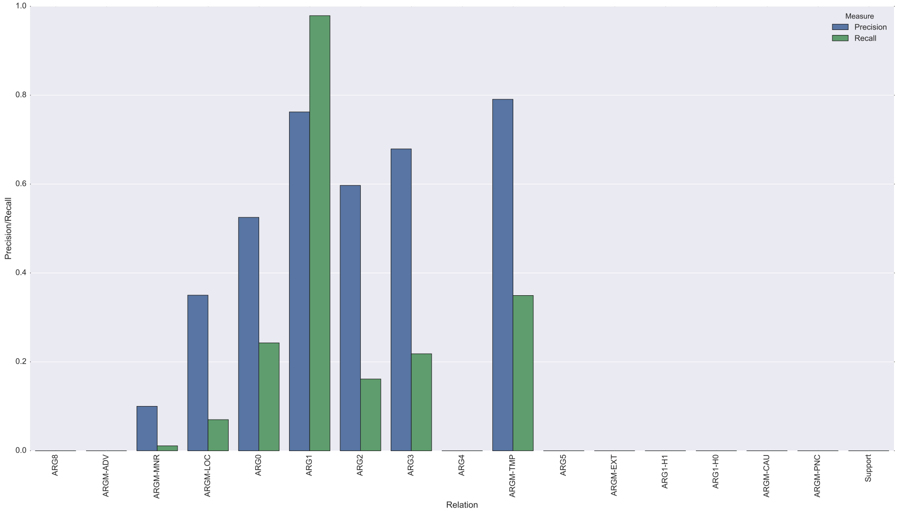
Table: 1-NN vs. k-NN. Other parameters: lemma, $300d$, Wiki+Giga, PAIRDIRECTION

Clear increase in accuracy, but . . .

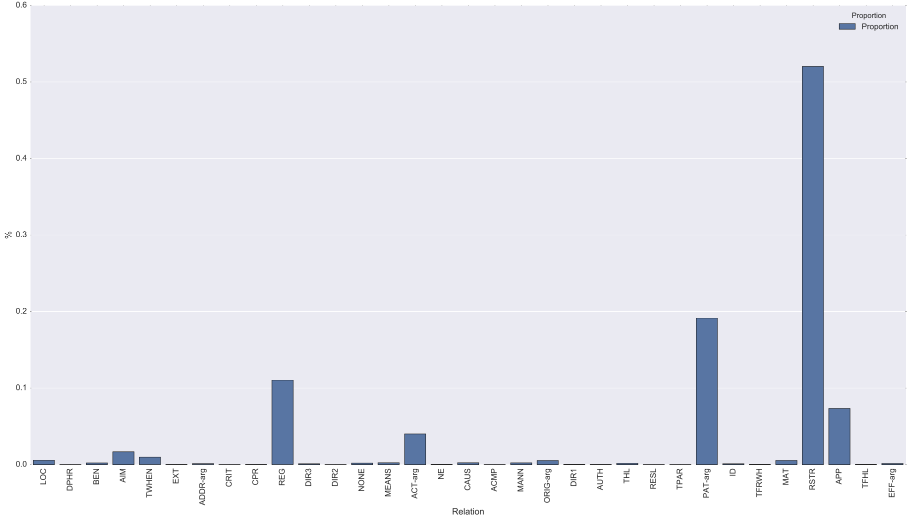
Nombank: Distribution of Relations



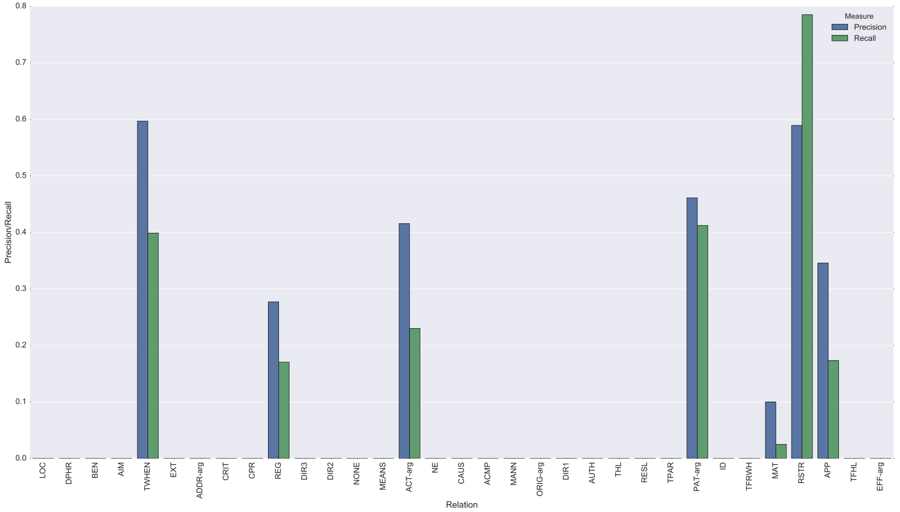
Per Relation Precision & Recall - Nombank



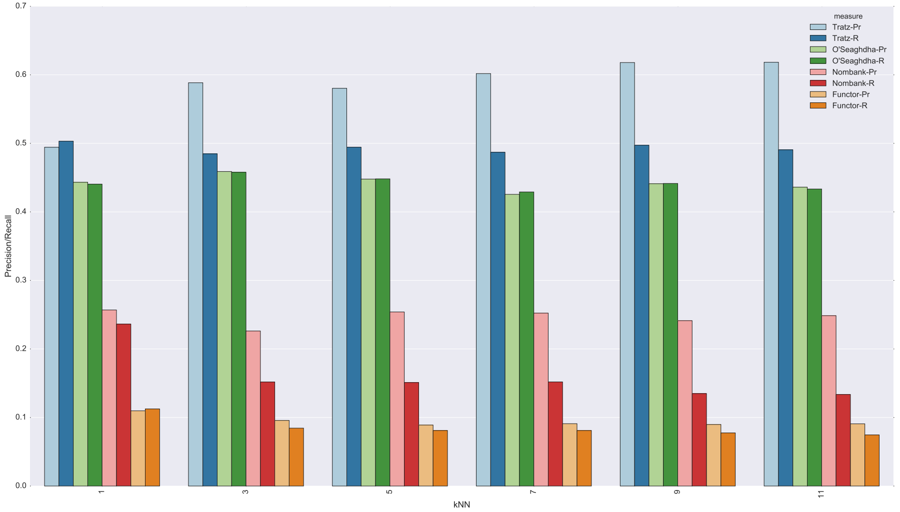
Functor: Distribution of Relations



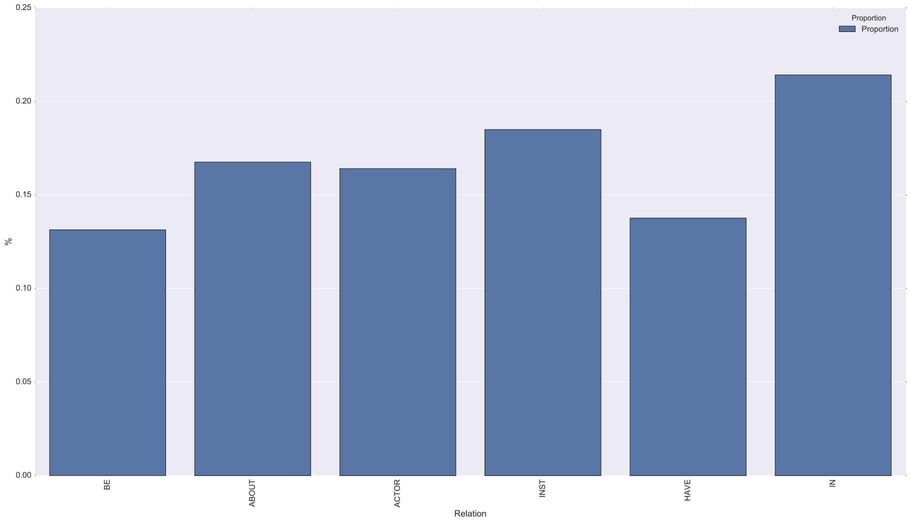
Per Relation Precision & Recall - Functor



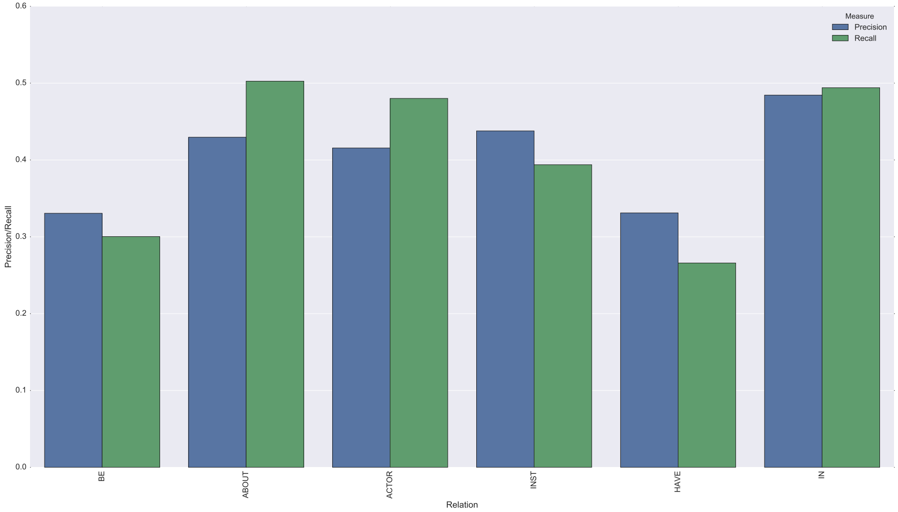
Macro-average: Precision & Recall



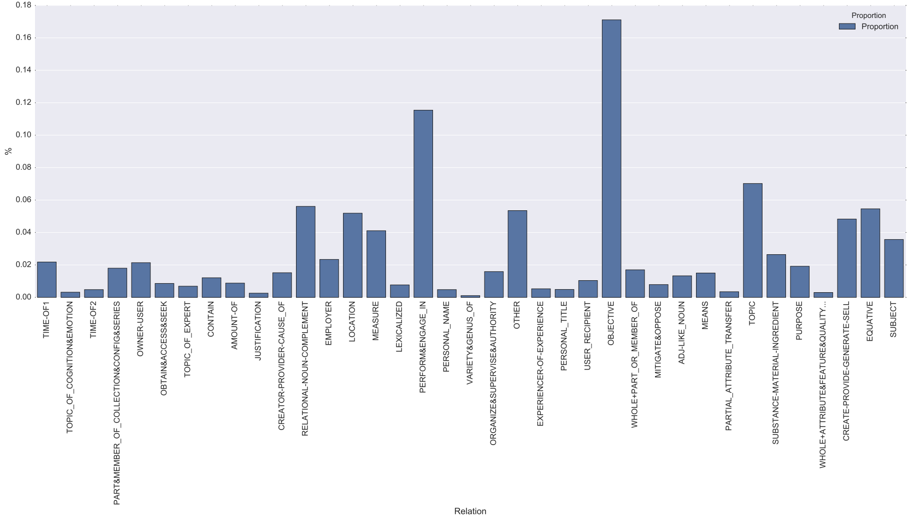
Ó Séaghdha: Distribution of Relations



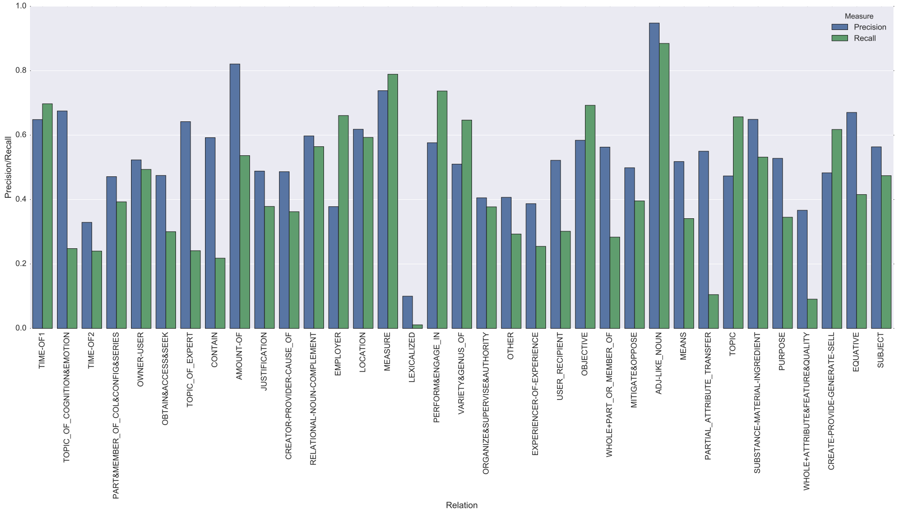
Per Relation Precision & Recall - Ó Séaghdha



Tratz: Distribution of Relations



Per Relation Precision & Recall - Tratz



Closer Look into Tratz's Dataset

- ADJ-LIKE_NOUN has only 7 distinct modifiers for 254 compounds [Dima, 2016]
- AMOUNT_OF has only 15 distinct heads for 168 compounds [Dima, 2016]
- which means . . .

Lexical memorization not relation learning :-(

"Lexical Memorization is the phenomenon in which the classifier learns that a specific word in a specific slot is a strong indicator of the label." [Levy et al., 2015]

Closer Look into Tratz's Dataset

- `ADJ-LIKE_NOUN` has only 7 distinct modifiers for 254 compounds [Dima, 2016]
- `AMOUNT_OF` has only 15 distinct heads for 168 compounds [Dima, 2016]
- which means . . .

Lexical memorization not relation learning :-(

"Lexical Memorization is the phenomenon in which the classifier learns that a specific word in a specific slot is a strong indicator of the label." [Levy et al., 2015]

Recap: Putting the Results in Perspective

	Tratz	Ó Séaghdha	Nombank	Functor
Majority-class baseline	17	15.4	71	51.7
State-of-the-art	79.3	63.1	n/a	n/a
$n = 11$	61.35	44.72	72.18	51.7

Recommendations to Train GloVe Models

- Use lemmas (if possible)
- Use the original C implementation of GloVe
- Start with 300d vectors
- More training data is not necessarily better
- Pre-shuffle the training data to make your experiments 'more' deterministic

Recommendations to Train GloVe Models

- Use lemmas (if possible)
- Use the original C implementation of GloVe
- Start with 300d vectors
- More training data is not necessarily better
- Pre-shuffle the training data to make your experiments ‘more’ deterministic

Recommendations to Train GloVe Models

- Use lemmas (if possible)
- Use the original C implementation of GloVe
- Start with 300d vectors
- More training data is not necessarily better
- Pre-shuffle the training data to make your experiments 'more' deterministic

Recommendations to Train GloVe Models

- Use lemmas (if possible)
- Use the original C implementation of GloVe
- Start with 300d vectors
- More training data is not necessarily better
- Pre-shuffle the training data to make your experiments ‘more’ deterministic

Recommendations to Train GloVe Models

- Use lemmas (if possible)
- Use the original C implementation of GloVe
- Start with 300d vectors
- More training data is not necessarily better
- Pre-shuffle the training data to make your experiments ‘more’ deterministic

References I

Baldwin, T. and Tanaka, T. (2004).
Translation by machine of complex nominals. Getting it right.
In *Second ACL Workshop on Multiword Expressions: Integrating Processing*, page 24 – 31, Barcelona, Spain.

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

References II

Dima, C. and Hinrichs, E. (2015).
Automatic Noun Compound Interpretation using Deep Neural Networks and Word Embeddings.
In *Proceedings of the 11th International Conference on Computational Semantics*, pages 173–183, London, UK. Association for Computational Linguistics.

Downing, P. (1977).
On the creation and use of English compound nouns.
Language, 53(4):810 – 842.

Girju, R., Moldovan, D., Tatu, M., and Antohe, D. (2005).
On the semantics of noun compounds.
Computer Speech & Language, 19(4):479 – 496.

References III

Lauer, M. and Dras, M. (1994).
A probabilistic model of compound nouns.
In *Proceedings of the 7th Australian Joint Conference on AI*.

Levy, O. and Goldberg, Y. (2014).
Linguistic Regularities in Sparse and Explicit Word Representations.
In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180, Ann Arbor, Michigan. Association for Computational Linguistics.

References IV

Levy, O., Remus, S., Biemann, C., and Dagan, I. (2015).
Do Supervised Distributional Methods Really Learn Lexical Inference Relations?
In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado. Association for Computational Linguistics.

Li, C. N. (1971).
Semantics and the Structure of Compounds in Chinese.
PhD thesis, University of California, Berkeley.

References V

Mikolov, T., Yih, W.-t., and Zweig, G. (2013).
Linguistic Regularities in Continuous Space Word Representations.
In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Nakov, P. (2013).
On the Interpretation of Noun Compounds: Syntax, Semantics, and Entailment.
Natural Language Engineering, 19(3):291–330.

References VI

Ó Séaghdha, D. (2008).
Learning compound noun semantics.
Technical Report UCAM-CL-TR-735, University of Cambridge, Computer Laboratory, Cambridge, UK.

Ó Séaghdha, D. and Copestake, A. (2009).
Using Lexical and Relational Similarity to Classify Semantic Relations.
In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 621–629, Athens, Greece. Association for Computational Linguistics.

Ó Séaghdha, D. and Copestake, A. (2013).
Interpreting compound nouns with kernel methods.
Journal of Natural Language Engineering, 19(3):331 – 356.

References VII

Pennington, J., Socher, R., and Manning, C. D. (2014).
GloVe: Global Vectors for Word Representation.
In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Tratz, S. and Hovy, E. (2010).
A taxonomy, dataset, and classifier for automatic noun compound interpretation.
In *Proceedings of the 48th Meeting of the Association for Computational Linguistics*, page 678 – 687, Uppsala, Sweden.