



Syntactic Parsing for Ranking-Based Coreference Resolution

Altaf Rahman and Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas

Noun Phrase Coreference

- Identify all noun phrases (NPs) that refer to the same entity

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

Noun Phrase Coreference

- Identify all noun phrases (NPs) that refer to the same entity

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

Noun Phrase Coreference

- Identify all noun phrases (NPs) that refer to the same entity

Queen Elizabeth set about transforming her **husband**,
King George VI, into a viable monarch.

Noun Phrase Coreference

- Identify all noun phrases (NPs) that refer to the same entity

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

Noun Phrase Coreference

- Identify all noun phrases (NPs) that refer to the same entity

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

Goal

- Employ linguistic features derived from syntactic parse trees to improve learning-based coreference resolution systems

Goal

- Employ linguistic features derived from syntactic parse trees to improve learning-based coreference resolution systems
- But ... there has been extensive prior work on using syntactic features for coreference resolution
 - Binding Constraints
 - Syntactic salience
 - ...

Goal

- Employ two types of parse-based features to improve learning-based coreference resolution systems
 - path-based features
 - tree-based features

Goal

- Employ two types of parse-based features to improve learning-based coreference resolution systems
 - path-based features
 - paths derived from parse trees
 - tree-based features

Goal

- Employ two types of parse-based features to improve learning-based coreference resolution systems
 - path-based features
 - paths derived from parse trees
 - tree-based features
 - trees as **structured** features
 - rather than design heuristics to extract features from a parse tree, use the tree itself **directly** as a feature

Goal

- Employ two types of parse-based features to improve learning-based coreference resolution systems
 - path-based features
 - paths derived from parse trees
 - tree-based features
 - trees as **structured** features
 - rather than design heuristics to extract features from a parse tree, use the tree itself **directly** as a feature
- But ... there has been work on using structured features to train an SVM for coreference resolution
 - Yang et al. (2006), Versley et al. (2008), Zhou & Kong (2009)

So, what's new?

- To understand the contributions of our work, we need to first understand the current state of coreference research

The Standard Approach to Coreference

- Process each NP in a text in a left-to-right manner.
- For each NP encountered, perform **2 steps**:
 1. determine whether the NP has an antecedent
 2. if so, identify an antecedent for it

The Standard Approach to Coreference

- Process each NP in a text in a left-to-right manner.
- For each NP encountered, perform **2 steps**:
 1. determine whether the NP has an antecedent
(Anaphoricity Determination)
 2. if so, identify an antecedent for it

The Standard Approach to Coreference

- Process each NP in a text in a left-to-right manner.
- For each NP encountered, perform **2 steps**:
 1. determine whether the NP has an antecedent
(Anaphoricity Determination)
 2. if so, identify an antecedent for it
(Antecedent Selection)

The Standard Approach to Coreference

- Both steps have been implemented using **machine learning**
- For **antecedent selection**,
 - numerous supervised coreference models have been designed
 - the most commonly used model: the **mention-pair model**

Mention-Pair Model

- a classifier that determines whether two NPs are coreferent
- Each training instance corresponds to two NPs

Mention-Pair Model

- a classifier that determines whether two NPs are coreferent
- Each training instance corresponds to two NPs

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

Mention-Pair Model

- a classifier that determines whether two NPs are coreferent
- Each training instance corresponds to two NPs

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

(her, Queen Elizabeth, ✓)

Mention-Pair Model

- a classifier that determines whether two NPs are coreferent
- Each training instance corresponds to two NPs

Queen Elizabeth set about transforming her husband.

King George VI, into a viable monarch.

(her, Queen Elizabeth, ✓)

(husband, Queen Elizabeth, ✗)

Mention-Pair Model

- a classifier that determines whether two NPs are coreferent
- Each training instance corresponds to two NPs

Queen Elizabeth set about transforming **her** husband.

King George VI, into a viable monarch.

(her, Queen Elizabeth, ✓)

(husband, Queen Elizabeth, ✗)

(husband, her, ✗)

The Mention-Pair Model is Weak

- **Limited expressiveness**
 - information extracted from two NPs may not be sufficient for making an informed coreference decision
- **Can't determine which candidate antecedent is the best**
 - only determine how good a candidate is relative to NP to be resolved, not how good it is relative to the others

How to Improve Model Expressiveness?

- Train a classifier that determines whether an NP belongs to a preceding coreference cluster
- Each training instance corresponds to an NP and a preceding cluster of NPs

How to Improve Model Expressiveness?

- Train a classifier that determines whether an NP belongs to a preceding coreference cluster
- Each training instance corresponds to an NP and a preceding cluster of NPs

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

How to Improve Model Expressiveness?

- Train a classifier that determines whether an NP belongs to a preceding coreference cluster
- Each training instance corresponds to an NP and a preceding cluster of NPs

Queen Elizabeth set about transforming her husband, King George VI, into a viable monarch.

(her, [Queen Elizabeth], ✓)

How to Improve Model Expressiveness?

- Train a classifier that determines whether an NP belongs to a preceding coreference cluster
- Each training instance corresponds to an NP and a preceding cluster of NPs

Queen Elizabeth set about transforming her husband.

King George VI, into a viable monarch.

(her, [Queen Elizabeth], ✓)

(husband, [Queen Elizabeth, her], ✗)

How to Improve Model Expressiveness?

- Train a classifier that determines whether an NP belongs to a preceding coreference cluster
- Each training instance corresponds to an NP and a preceding cluster of NPs

Queen Elizabeth set about transforming her husband,

King George VI, into a viable monarch.

(her, [Queen Elizabeth], ✓)

(husband, [Queen Elizabeth, her], ✗)

(King George VI, [Queen Elizabeth, her], ✗)

(King George VI, [husband], ✓)

How to Improve Model Expressiveness?

- This model is more **expressive** than the mention-pair model
 - can employ **cluster-level** features defined over any subset of NPs in a preceding cluster
- But ... it does not address the problem of the model's failure to compare candidate antecedents and identify the best one

How to Identify the Best Antecedent?

- Train a model to impose a **ranking** on the candidate antecedents for an NP to be resolved
 - it assigns the highest rank to the correct antecedent
- Each training instance corresponds to an NP to be resolved and one of its candidate antecedents

How to Identify the Best Antecedent?

- Train a model to impose a **ranking** on the candidate antecedents for an NP to be resolved
 - it assigns the highest rank to the correct antecedent
- Each training instance corresponds to an NP to be resolved and one of its candidate antecedents

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

How to Identify the Best Antecedent?

- Train a model to impose a **ranking** on the candidate antecedents for an NP to be resolved
 - it assigns the highest rank to the correct antecedent
- Each training instance corresponds to an NP to be resolved and one of its candidate antecedents

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

“her” has only one candidate antecedent; nothing to rank

How to Identify the Best Antecedent?

- Train a model to impose a **ranking** on the candidate antecedents for an NP to be resolved
 - it assigns the highest rank to the correct antecedent
- Each training instance corresponds to an NP to be resolved and one of its candidate antecedents

Queen Elizabeth set about transforming her husband.

King George VI, into a viable monarch.

“husband” has two candidate antecedents with the same rank;
so nothing to rank

How to Identify the Best Antecedent?

- Train a model to impose a **ranking** on the candidate antecedents for an NP to be resolved
 - it assigns the highest rank to the correct antecedent
- Each training instance corresponds to an NP to be resolved and one of its candidate antecedents

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

“King George VI” has three candidate antecedents, has something to rank, so generate three training instances:
(King George VI, Queen Elizabeth, **low**)
(King George VI, her, **low**)
(King George VI, husband, **high**)

How to Identify the Best Antecedent?

- Train a model to impose a **ranking** on the candidate antecedents for an NP to be resolved
 - it assigns the highest rank to the correct antecedent
- Each training instance corresponds to an NP to be resolved and one of its candidate antecedents

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

“King George VI” has three candidate antecedents, has something to rank, so generate three training instances:

(King George VI, Queen Elizabeth, **low**)

(King George VI, her, **low**)

(King George VI, husband, **high**)

This constitutes **one** ranking problem

How to Identify the Best Antecedent?

- Train a model to impose a **ranking** on the candidate antecedents for an NP to be resolved
 - it assigns the highest rank to the correct antecedent
- Each training instance corresponds to an NP to be resolved and one of its candidate antecedents

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

“King George VI” has three candidate antecedents. To solve something to rank, so generate three training instances:
(King George VI, Queen Elizabeth, **low**)
(King George VI, her, **low**)
(King George VI, husband, **high**)

A learner will learn to compare all candidate antecedents in each ranking problem in the training set

How to Identify the Best Antecedent?

- addresses the problem of identifying the best candidate antecedent
- But ... it does not address the expressiveness problem

So ...

- To combine the best of both worlds, we train a **ranker that ranks preceding clusters**, not candidate antecedents

Cluster-Ranking Model

- A **ranker** trained to rank preceding clusters
- Each training instance corresponds to an NP to be resolved and a preceding cluster

Rahman & Ng (2009)

Cluster-Ranking Model

- A **ranker** trained to rank preceding clusters
- Each training instance corresponds to an NP to be resolved and a preceding cluster

Queen Elizabeth set about transforming her husband, King George VI, into a viable monarch.

Has only one preceding cluster; nothing to rank

Rahman & Ng (2009)

Cluster-Ranking Model

- A **ranker** trained to rank preceding clusters
- Each training instance corresponds to an NP to be resolved and a preceding cluster

Queen Elizabeth set about transforming her husband.

King George VI, into a viable monarch.

“husband” has one preceding cluster, [Queen Elizabeth, her],
so nothing to rank

Rahman & Ng (2009)

Cluster-Ranking Model

- A **ranker** trained to rank preceding clusters
- Each training instance corresponds to an NP to be resolved and a preceding cluster

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

“King George VI” has two preceding clusters, has something to rank, so generate two instances:

(King George VI, [Queen Elizabeth, her], **low**)

(King George VI, [husband], **high**)

Cluster-Ranking Model

- A **ranker** trained to rank preceding clusters
- Each training instance corresponds to an NP to be resolved and a preceding cluster

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

“King George VI” has two preceding clusters, has something to rank, so generate two instances:

(King George VI, [Queen Elizabeth, her], **low**)

(King George VI, [husband], **high**)

This constitutes **one** ranking problem

An Improvement to Cluster-Ranking Model

- Observation
 - In the standard approach to coreference, anaphoricity determination is performed prior to antecedent selection

An Improvement to Cluster-Ranking Model

- Observation
 - In the standard approach to coreference, anaphoricity determination is performed prior to antecedent selection
- Weakness of this pipeline architecture
 - Errors in anaphoricity determination will propagate to the antecedent selection component

An Improvement to Cluster-Ranking Model

- Observation
 - In the standard approach to coreference, anaphoricity determination is performed prior to antecedent selection
- Weakness of this pipeline architecture
 - Errors in anaphoricity determination will propagate to the antecedent selection component
- This weakness can be addressed by jointly learning anaphoricity determination and antecedent selection

Joint Learning for Anaphoricity Determination and Antecedent Selection

- Need to ensure that the ranker is given the **option** to determine an NP as not having an antecedent

Joint Learning for Anaphoricity Determination and Antecedent Selection

- Need to ensure that the ranker is given the **option** to determine an NP as not having an antecedent
 - Easy. Simply create an **additional** instance for each ranking problem that corresponds to the “null” cluster
 - Selecting the “null” cluster amounts to determining that an NP does **not** have an antecedent

Joint Learning for Anaphoricity

Determination and Antecedent Selection

- Need to ensure that the ranker is given the **option** to determine an NP as not having an antecedent
 - Easy. Simply create an **additional** instance for each ranking problem that corresponds to the “null” cluster
 - Selecting the “null” cluster amounts to determining that an NP does **not** have an antecedent

Queen Elizabeth set about transforming her husband,
King George VI into a viable monarch.

Has two preceding clusters, has something to rank, so generate two instances:

(King George VI, [Queen Elizabeth, her], **low**)

(King George VI, [husband], **high**)

Joint Learning for Anaphoricity Determination and Antecedent Selection

- Need to ensure that the ranker is given the **option** to determine an NP as not having an antecedent
 - Easy. Simply create an **additional** instance for each ranking problem that corresponds to the “null” cluster
 - Selecting the “null” cluster amounts to determining that an NP does **not** have an antecedent

Queen Elizabeth set about transforming her husband,
King George VI, into a viable monarch.

Has two preceding clusters, has something to rank, so generate two instances:

(King George VI, [Queen Elizabeth, her], **low**)

(King George VI, [husband], **high**)

(King George VI, null, **low**) ←

Joint Learning for Anaphoricity Determination and Antecedent Selection

- Incorporating joint learning into the cluster-ranking model yields the **joint cluster-ranking model**
 - a **state-of-the-art** supervised coreference model

Goal

- Employ path-based features and tree-based structured features to improve learning-based coreference systems

What's new?

- We use structured features to improve anaphoricity determination (in particular, to identify non-anaphoric NPs)
 - Prior work aims to use them to improve antecedent selection
- We use structured features to improve the joint cluster ranking model
 - Prior work aims to use them to improve the mention-pair model
 - We know how to employ structured features to train a classifier
 - but ... it's not immediately clear how to do so in a ranking model

How to use Structured Features in the Mention-Pair Model?

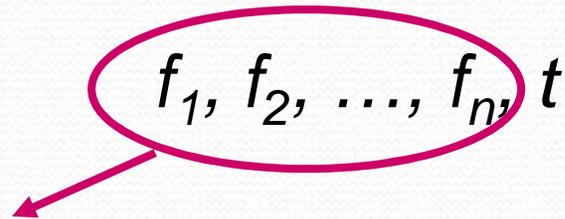
How to use Structured Features in the Mention-Pair Model?

- Each instance corresponds to two NPs, and has features:

$$f_1, f_2, \dots, f_n, t$$

How to use Structured Features in the Mention-Pair Model?

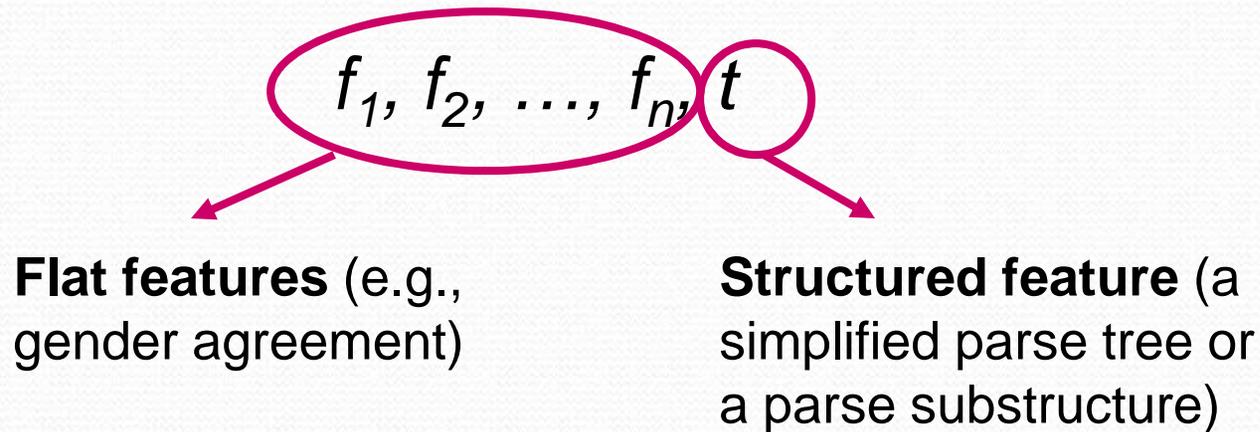
- Each instance corresponds to two NPs, and has features:



Flat features (e.g.,
gender agreement)

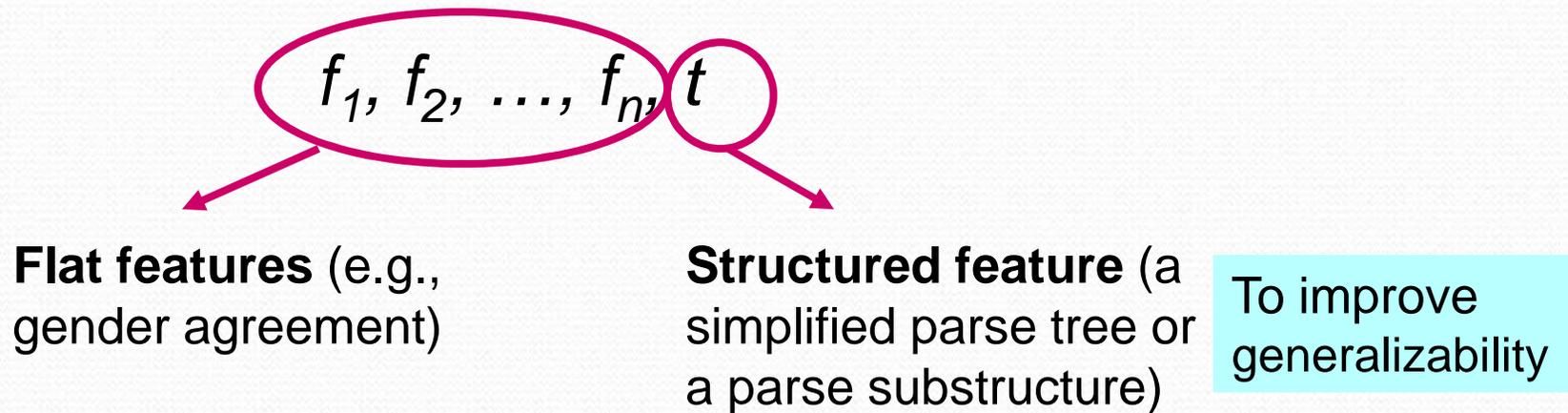
How to use Structured Features in the Mention-Pair Model?

- Each instance corresponds to two NPs, and has features:



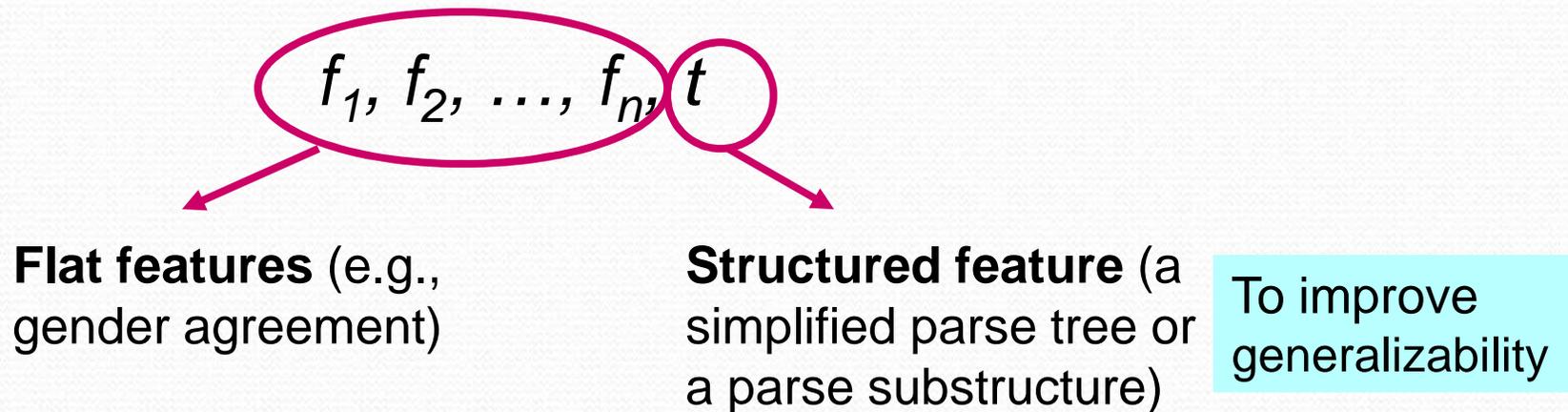
How to use Structured Features in the Mention-Pair Model?

- Each instance corresponds to two NPs, and has features:



How to use Structured Features in the Mention-Pair Model?

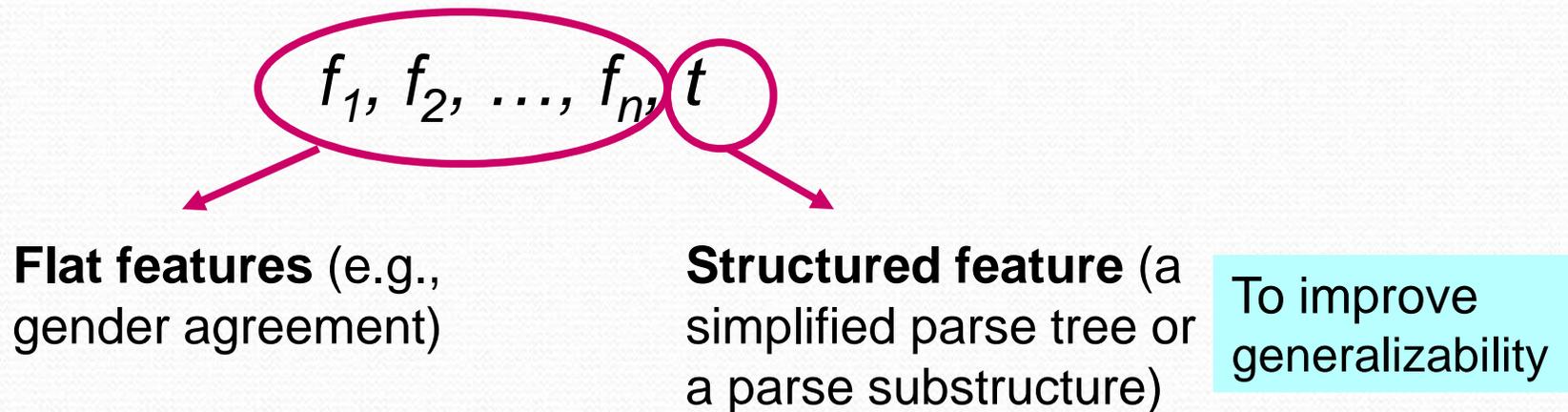
- Each instance corresponds to two NPs, and has features:



- Step 1: Compute the similarity between two instances
 - Compute similarity over their flat features (using a **linear** kernel)
 - Compute similarity over their trees (using a **tree** kernel)
 - Combine the two similarity values using a **composite** kernel

How to use Structured Features in the Mention-Pair Model?

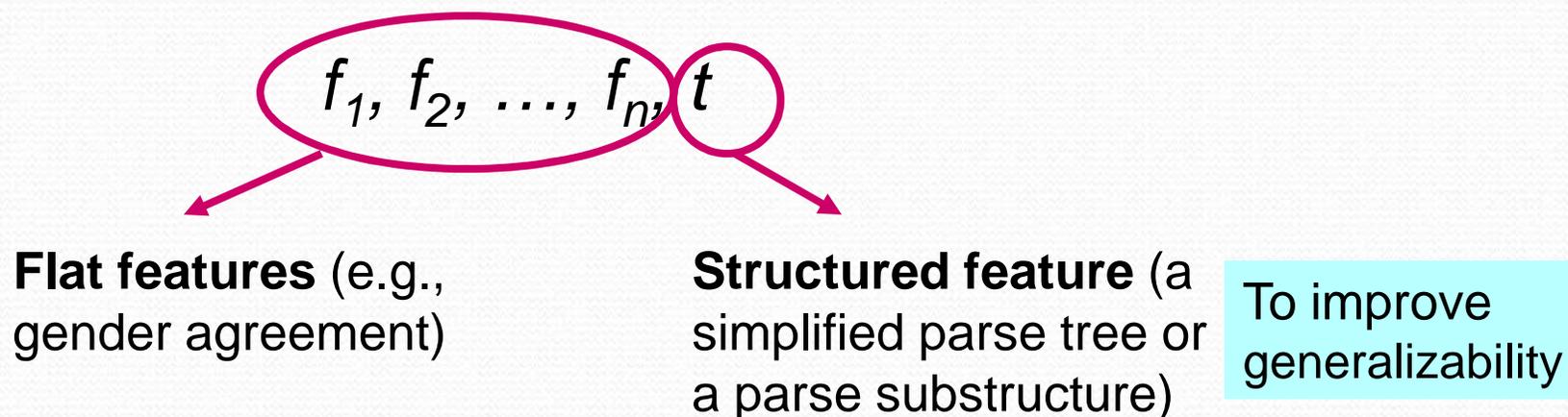
- Each instance corresponds to two NPs, and has features:



- Step 1: Compute the similarity between two instances
 - Compute similarity over their flat features (using a **linear** kernel)
 - Compute similarity over their trees (using a **tree** kernel)
 - Combine the two similarity values using a **composite** kernel
- Step 2: Learn using an off-the-shelf SVM learner

How to use Structured Features in the Cluster-Ranking Model?

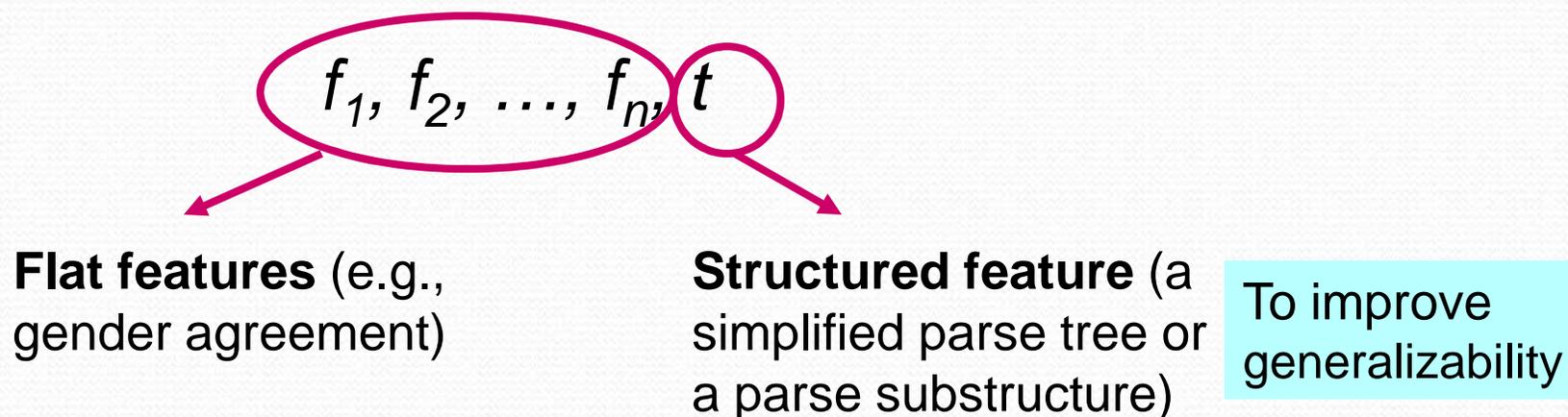
- Each instance corresponds to two NPs, and has features:



- Step 1: Compute the similarity between two instances
 - Compute similarity over their flat features (using a **linear** kernel)
 - Compute similarity over their trees (using a **tree** kernel)
 - Combine the two similarity values using a **composite** kernel
- Step 2: Learn using an off-the-shelf SVM learner

How to use Structured Features in the Cluster-Ranking Model?

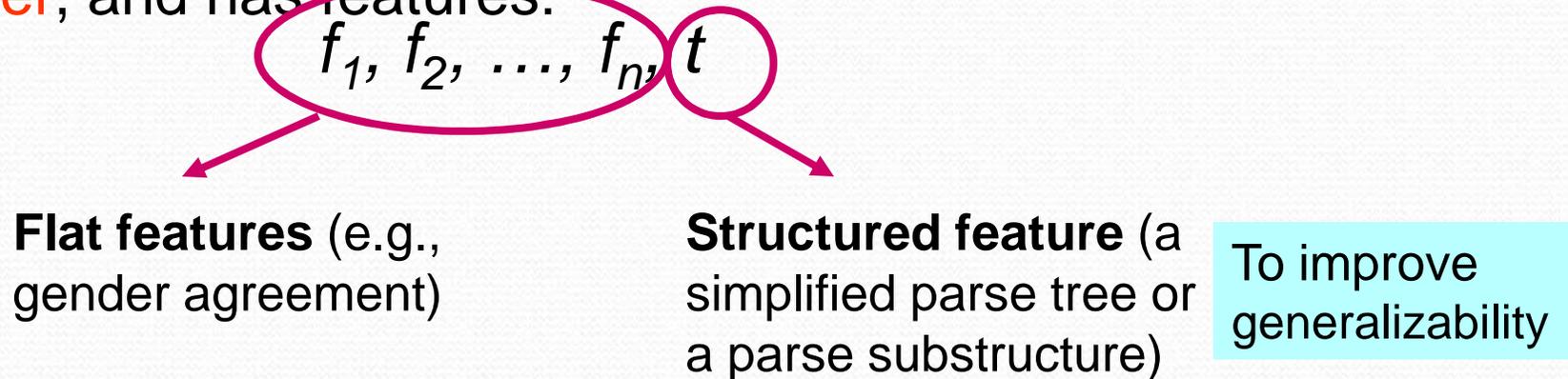
- Each instance corresponds to ~~two NPs~~, and has features:



- Step 1: Compute the similarity between two instances
 - Compute similarity over their flat features (using a **linear** kernel)
 - Compute similarity over their trees (using a **tree** kernel)
 - Combine the two similarity values using a **composite** kernel
- Step 2: Learn using an off-the-shelf SVM learner

How to use Structured Features in the Cluster-Ranking Model?

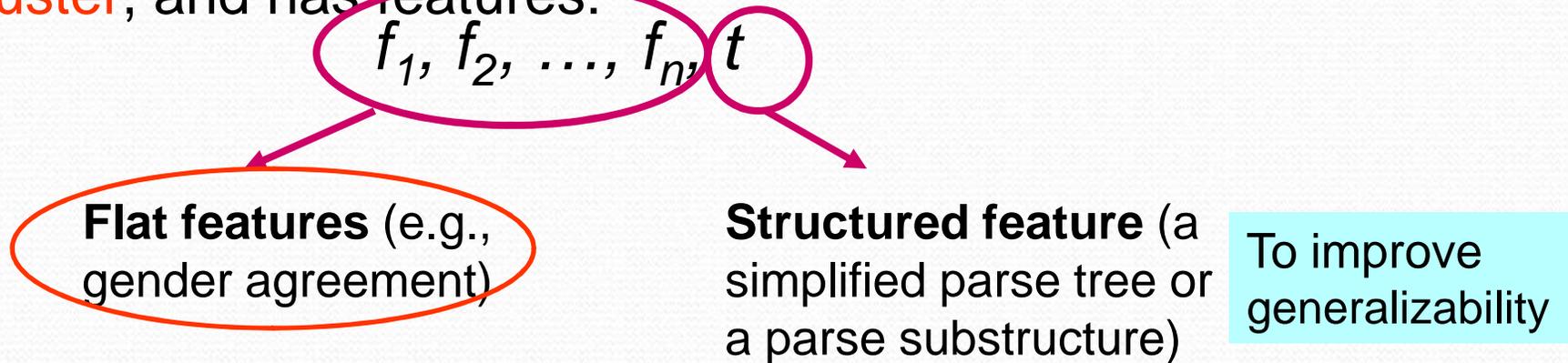
- Each instance corresponds to an NP and its preceding cluster, and has features.



- Step 1: Compute the similarity between two instances
 - Compute similarity over their flat features (using a **linear** kernel)
 - Compute similarity over their trees (using a **tree** kernel)
 - Combine the two similarity values using a **composite** kernel
- Step 2: Learn using an off-the-shelf SVM learner

How to use Structured Features in the Cluster-Ranking Model?

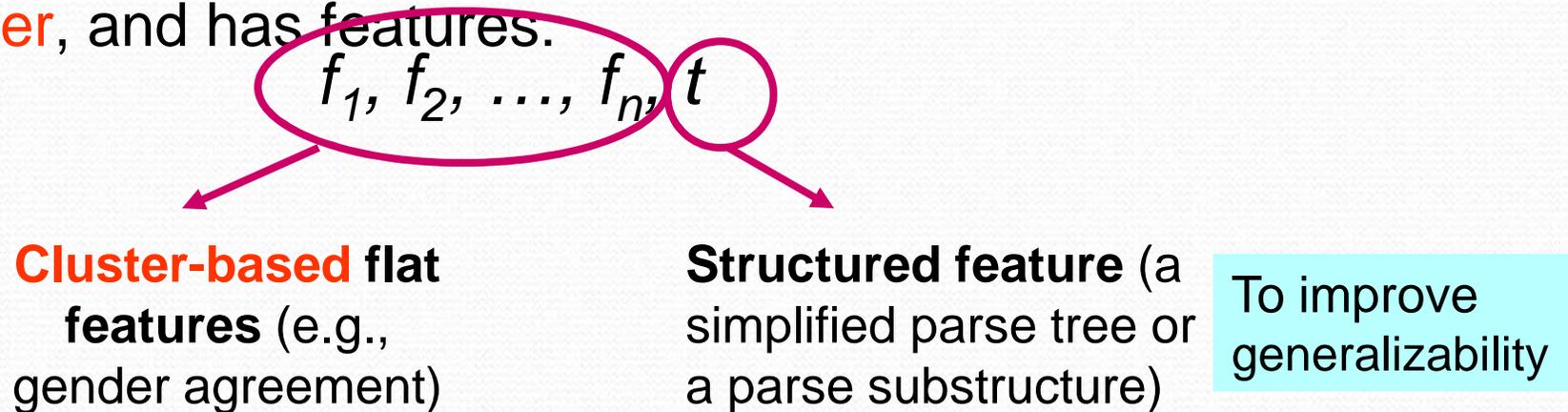
- Each instance corresponds to an NP and its preceding cluster, and has features.



- Step 1: Compute the similarity between two instances
 - Compute similarity over their flat features (using a **linear** kernel)
 - Compute similarity over their trees (using a **tree** kernel)
 - Combine the two similarity values using a **composite** kernel
- Step 2: Learn using an off-the-shelf SVM learner

How to use Structured Features in the Cluster-Ranking Model?

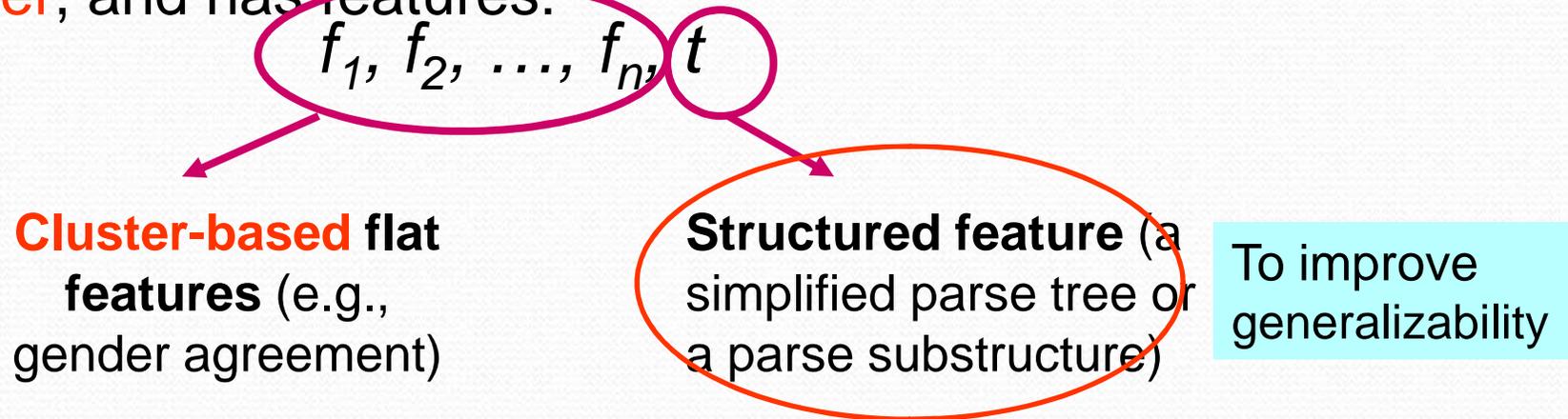
- Each instance corresponds to an NP and its preceding cluster, and has features.



- Step 1: Compute the similarity between two instances
 - Compute similarity over their flat features (using a **linear** kernel)
 - Compute similarity over their trees (using a **tree** kernel)
 - Combine the two similarity values using a **composite** kernel
- Step 2: Learn using an off-the-shelf SVM learner

How to use Structured Features in the Cluster-Ranking Model?

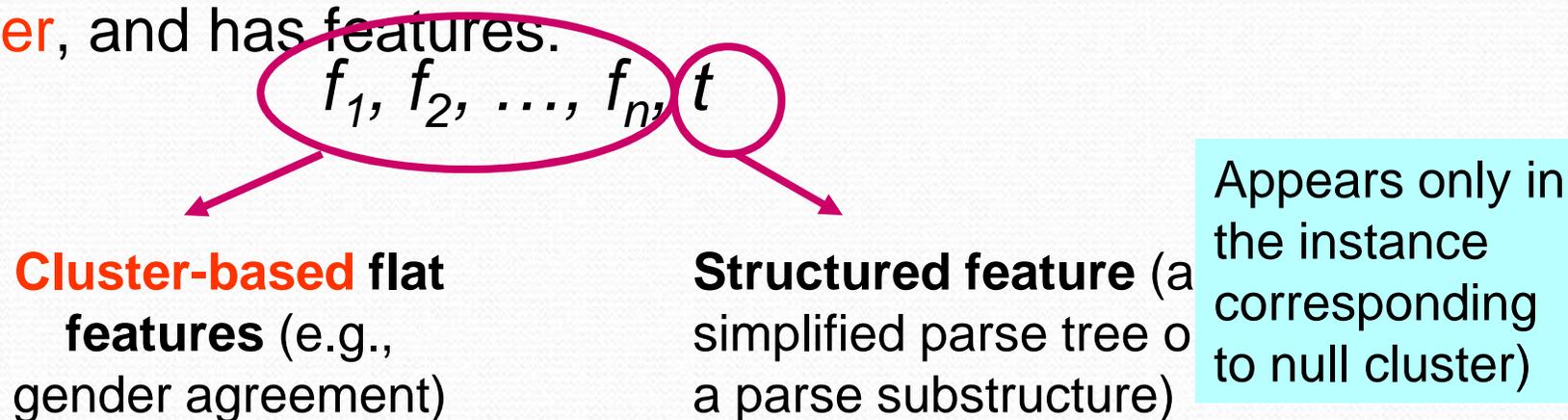
- Each instance corresponds to an NP and its preceding cluster, and has features.



- Step 1: Compute the similarity between two instances
 - Compute similarity over their flat features (using a **linear** kernel)
 - Compute similarity over their trees (using a **tree** kernel)
 - Combine the two similarity values using a **composite** kernel
- Step 2: Learn using an off-the-shelf SVM learner

How to use Structured Features in the Cluster-Ranking Model?

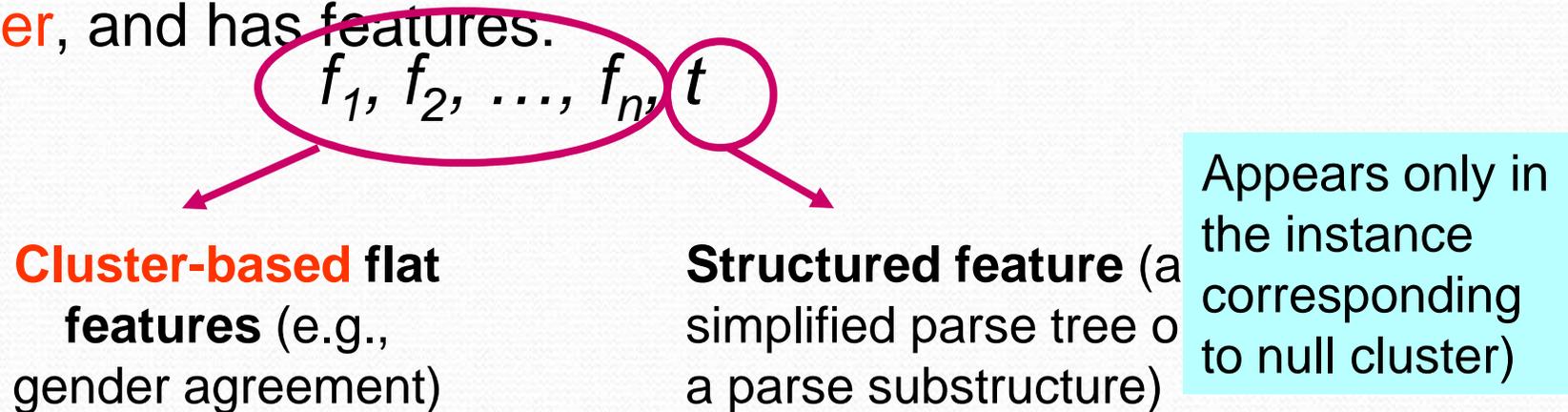
- Each instance corresponds to an NP and its preceding cluster, and has features.



- Step 1: Compute the similarity between two instances
 - Compute similarity over their flat features (using a **linear** kernel)
 - Compute similarity over their trees (using a **tree** kernel)
 - Combine the two similarity values using a **composite** kernel
- Step 2: Learn using an off-the-shelf SVM learner

How to use Structured Features in the Cluster-Ranking Model?

- Each instance corresponds to an NP and its preceding cluster, and has features.



- Step 0: Recast ranking as classification**
- Step 1: Compute the similarity between two instances
 - Compute similarity over their flat features (using a **linear** kernel)
 - Compute similarity over their trees (using a **tree** kernel)
 - Combine the two similarity values using a **composite** kernel
- Step 2: Learn using an off-the-shelf SVM learner

Recasting Ranking as Classification

- Idea: convert the problem of ranking m objects into a set of pairwise ranking problems

Recasting Ranking as Classification

- Idea: convert the problem of ranking m objects into a set of pairwise ranking problems
 - Train a model that ranks two objects (in our case, two preceding clusters) at a time

Recasting Ranking as Classification

- Idea: convert the problem of ranking m objects into a set of pairwise ranking problems
 - Train a model that ranks two objects (in our case, two preceding clusters) at a time
 - Pairwise ranking is essentially a binary classification problem

The story so far ...

- We have talked about how to incorporate tree-based (structured) features into the cluster-ranking model
- We haven't talked about path-based features ...

What is a Path-Based Feature?

- **encodes** the **contextual relationship** between an NP to be resolved and a candidate antecedent
- **represented** as **the shortest sequence of nodes in the parse tree** that need to be traversed in order to reach the candidate antecedent from the NP to be resolved
- If the NP to be resolved and its candidate antecedent are in different sentences, we create an additional “root” node connecting the parse trees of the sentences they reside in

Path-Based Features (Cont')

- include in the feature set only those path-based features seen at least seven times in the training set
- Given an instance involving an NP and a preceding cluster, the value of a path-based feature is 1 if the path between the NP and any of the NPs in the preceding cluster is the same as the path represented by the feature
 - Otherwise, its feature value is 0

Evaluation

- Goal
 - Evaluate the effectiveness of path-based and tree-based (structured) features in improving the cluster-ranking model

Experimental Setup

- Coreference data set
 - 147 Switchboard dialogues (Nissim et al., 2004)
 - 117 for training, 30 for test
- Baseline coreference systems
 - cluster-ranking model (Rahman & Ng, 2009)
 - mention-pair model (Soon et al., 2001)
 - employs 39 features
 - neither of them uses path-based and tree-based features
 - trained using SVM^{light}
- Use manually annotated NPs
- Scoring programs
 - B³ (Bagga & Baldwin, 1998), ϕ_3 -CEAF (Luo, 2005)

Baseline Systems: Results

	B³				CEAF	
	R	P	F	% err. red.	F	% err. red.
Baseline Mention-Pair model	78.1	61.6	69.1	---	62.8	---
Baseline Cluster-Ranking model	71.1	78.2	74.5	---	68.5	---

Baseline Systems: Results

	B³				CEAF	
	R	P	F	% err. red.	F	% err. red.
Baseline Mention-Pair model	78.1	61.6	69.1	---	62.8	---
Baseline Cluster-Ranking model	71.1	78.2	74.5	---	68.5	---

Baseline Systems: Results

	B³				CEAF	
	R	P	F	% err. red.	F	% err. red.
Baseline Mention-Pair model	78.1	61.6	69.1	---	62.8	---
Baseline Cluster-Ranking model	71.1	78.2	74.5	---	68.5	---

Baseline Systems: Results

	B³				CEAF	
	R	P	F	% err. red.	F	% err. red.
Baseline Mention-Pair model	78.1	61.6	69.1	---	62.8	---
Baseline Cluster-Ranking model	71.1	78.2	74.5	---	68.5	---

Baseline Systems: Results

	B³				CEAF	
	R	P	F	% err. red.	F	% err. red.
Baseline Mention-Pair model	78.1	61.6	69.1	---	62.8	---
Baseline Cluster-Ranking model	71.1	78.2	74.5	---	68.5	---

Baseline Systems: Results

	B³				CEAF	
	R	P	F	% err. red.	F	% err. red.
Baseline Mention-Pair model	78.1	61.6	69.1	---	62.8	---
Baseline Cluster-Ranking model	71.1	78.2	74.5	---	68.5	---

Baseline Systems: Results

	B³				CEAF	
	R	P	F	% err. red.	F	% err. red.
Baseline Mention-Pair model	78.1	61.6	69.1	---	62.8	---
Baseline Cluster-Ranking model	71.1	78.2	74.5	---	68.5	---

- The cluster-ranking model outperforms the mention-pair model
- Improvements via path-based and tree-based features, if any, will be measured with respect to the cluster-ranking baseline

Incorporating Path-Based Features

	B³				CEAF	
	R	P	F	% err. red.	F	% err. red.
Baseline Mention-Pair model	78.1	61.6	69.1	---	62.8	---
Baseline Cluster-Ranking model	71.1	78.2	74.5	---	68.5	---
Cluster ranking + paths	76.4	75.2	75.8	(5.1)	70.6	(6.7)

- F-measure increases by 1.3 (B³) and 2.1 (CEAF)★
- **% err. red.:** % of error reduction of a system relative to CR baseline
 - Relative error reduced by 5.1% (B³) and 6.7% (CEAF)

Incorporating Tree-Based Features

	B³				CEAF	
	R	P	F	% err. red.	F	% err. red.
Baseline Mention-Pair model	78.1	61.6	69.1	---	62.8	---
Baseline Cluster-Ranking model	71.1	78.2	74.5	---	68.5	---
Cluster ranking + paths	76.4	75.2	75.8	(5.1)	70.6	(6.7)
Cluster ranking + trees	75.1	76.0	75.5	(3.9)	70.4	(6.0)

- F-measure increases by 1.0 (B³) and 1.9 (CEAF)★

Incorporating Both Path-Based and Tree-Based Features

	B³				CEAF	
	R	P	F	% err. red.	F	% err. red.
Baseline Mention-Pair model	78.1	61.6	69.1	---	62.8	---
Baseline Cluster-Ranking model	71.1	78.2	74.5	---	68.5	---
Cluster ranking + paths	76.4	75.2	75.8	(5.1)	70.6	(6.7)
Cluster ranking + trees	75.1	76.0	75.5	(3.9)	70.4	(6.0)
Cluster ranking + paths + trees	76.6	76.8	76.7	(8.6)	72.2	(11.7)

- F-measure increases by 2.2 (B³) and 3.7 (CEAF) 
 - equivalent to an error reduction of 8.6% (B³) and 11.7% (CEAF)

Incorporating Both Path-Based and Tree-Based Features

	B³				CEAF	
	R	P	F	% err. red.	F	% err. red.
Baseline Mention-Pair model	78.1	61.6	69.1	---	62.8	---
Baseline Cluster-Ranking model	71.1	78.2	74.5	---	68.5	---
Cluster ranking + paths	76.4	75.2	75.8	(5.1)	70.6	(6.7)
Cluster ranking + trees	75.1	76.0	75.5	(3.9)	70.4	(6.0)
Cluster ranking + paths + trees	76.6	76.8	76.7	(8.6)	72.2	(11.7)

- F-measure increases by 2.2 (B³) and 3.7 (CEAF) 
 - equivalent to an error reduction of 8.6% (B³) and 11.7% (CEAF)
- Better results are obtained when the two types of features are applied in combination

Summary

- Examined the effectiveness of tree-based and path-based features in improving the joint cluster-ranking model
 - when they were applied in combination, we saw a reduction in relative error by 8.6-11.7% on Switchboard dialogues
- Enabled flat and structured features to be used simultaneously in a ranking model that employs joint learning