# Robust Question Answering via Sub-part Alignment

TAUR LAB

Jifan Chen and Greg Durrett

The University of Texas at Austin

# QA models are easy to fool

- Current QA models work well in-domain, but they're not broadly robust when facing challenge settings

# QA models are easy to fool

- Current QA models work well in-domain, but they're not broadly robust when facing challenge settings

A simple adversarial attack could fool the model:

**Question**: What day was Super Bowl 50 played on?

**Context**: Super Bowl 50 was an American football game to determine the champion of NFL … The game was played on February 7, 2016 …

# QA models are easy to fool

- Current QA models work well in-domain, but they're not broadly robust when facing challenge settings

A simple adversarial attack could fool the model:

**Question**: What day was Super Bowl 50 played on?

**Context**: Super Bowl 50 was an American football game to determine the champion of NFL … The game was played on February 7, 2016 …

Append

**Adversarial Context**: The Champ Bowl was played on the day of August 18, 1991

Adversarial attack by Jia and Liang 2017

# QA models are easy to fool

- Current QA models work well in-domain, but they're not broadly robust when facing challenge settings

A simple adversarial attack could fool the model:

**Question**: What day was Super Bowl 50 played on?

**Context**: Super Bowl 50 was an American football game to determine the champion of NFL … The game was played on February 7, 2016 …

Append

**Adversarial Context**: The Champ Bowl was played on the day of August 18, 1991

- It is hard to understand and control the behaviors of such black-box models

Adversarial attack by Jia and Liang 2017

# Make QA explicit

# Make QA explicit

Core idea: Verify if the whole question is answered: break the question into smaller units and find their counterparts in the context

- If all units are well-supported, we can trust the prediction
- If not, we can reject the prediction or place constraints to control the model

# Make QA explicit

Core idea: Verify if the whole question is answered: break the question into smaller units and find their counterparts in the context

- If all units are well-supported, we can trust the prediction
- If not, we can reject the prediction or place constraints to control the model

**Adversarial Context**: Super Bowl 50 was an American football game to determine the champion NFL … The game was played on February 7, 2016 … The Champ Bowl was played on the day of August 18, 1991

**Question**: What day was Super Bowl 50 played on?

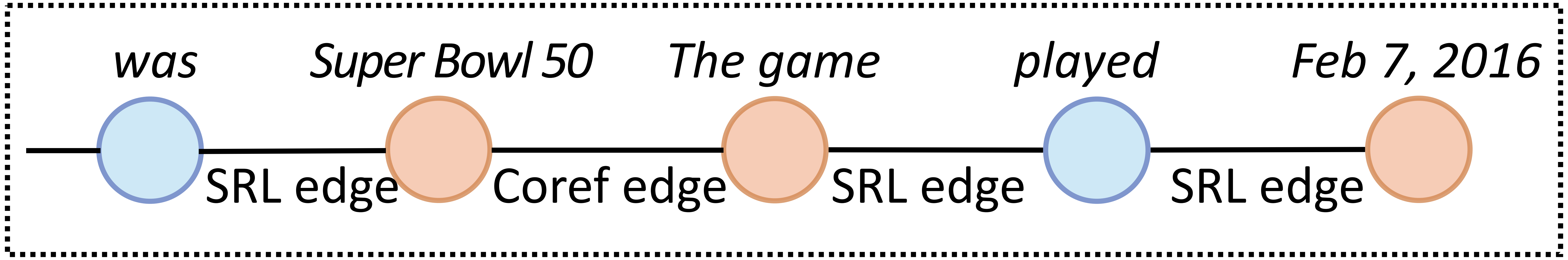
# Sub-part alignment for QA

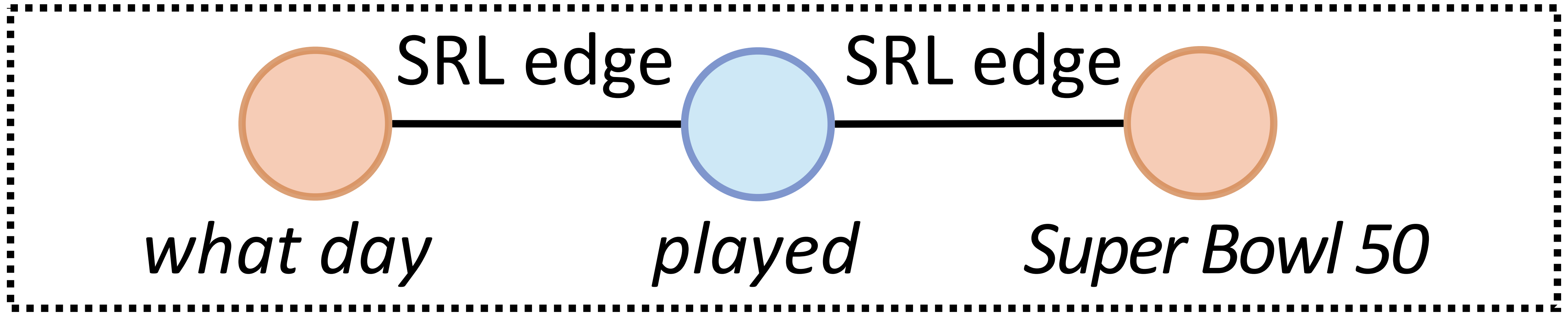
**Context:** Super Bowl 50 was … The game was played on Feb 7, 2016

**Question**: What day was Super Bowl 50 played on?

# Sub-part alignment for QA

**Context:** Super Bowl 50 was … The game was played on Feb 7, 2016



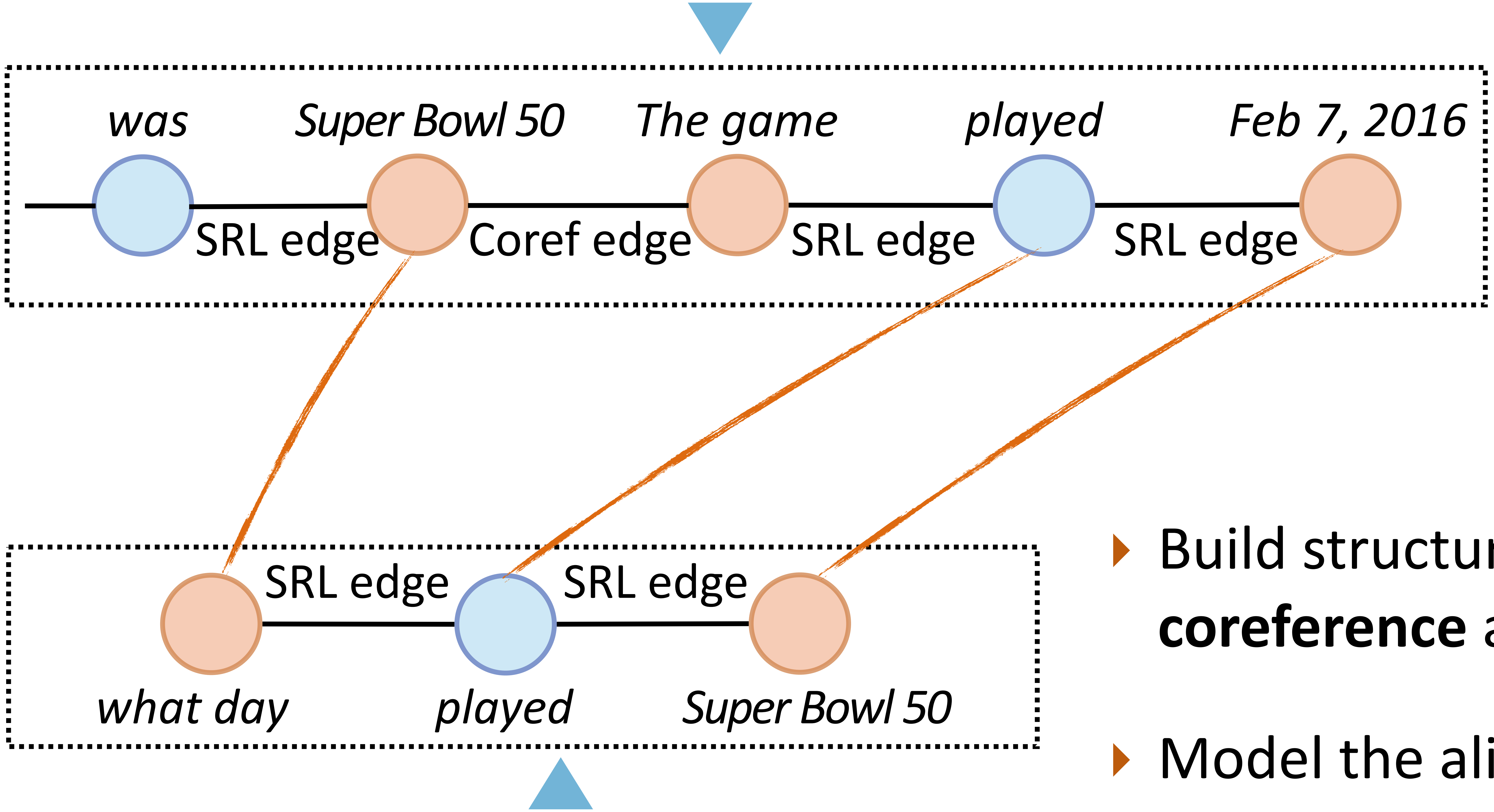




**Question**: What day was Super Bowl 50 played on?

- Build structured graph with **coreference** and **semantic role labeling**

# Sub-part alignment for QA

**Context:** Super Bowl 50 was … The game was played on Feb 7, 2016




**Question**: What day was Super Bowl 50 played on?

- Build structured graph with **coreference** and **semantic role labeling**
- Model the alignment between the question graph and the context graph

# Outline

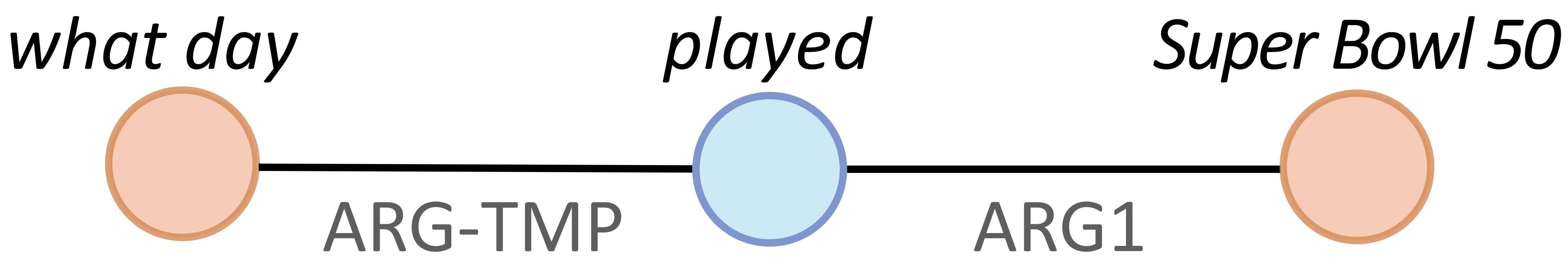
1) Question answering via sub-part alignment
   - Graph construction
   - Model: **graph alignment** between the question and the context
   - Inference: beam search respecting **constraints**
   - Training: **SSVM** using beam search

2) Experiments
   - Adversarial robustness
   - Constraints on alignment scores

3) Takeaways

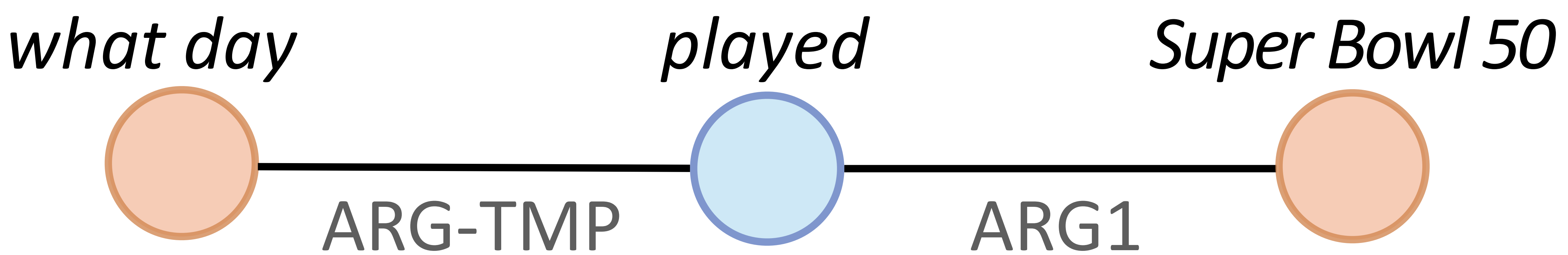
# Graph construction

**Question**: What day was Super Bowl 50 played on?

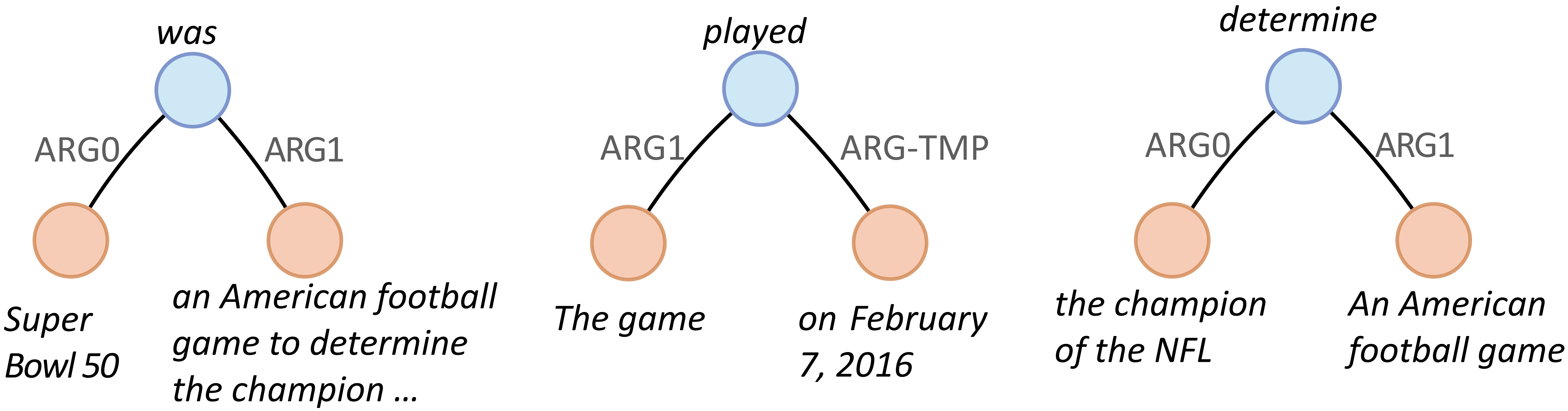


# Graph construction

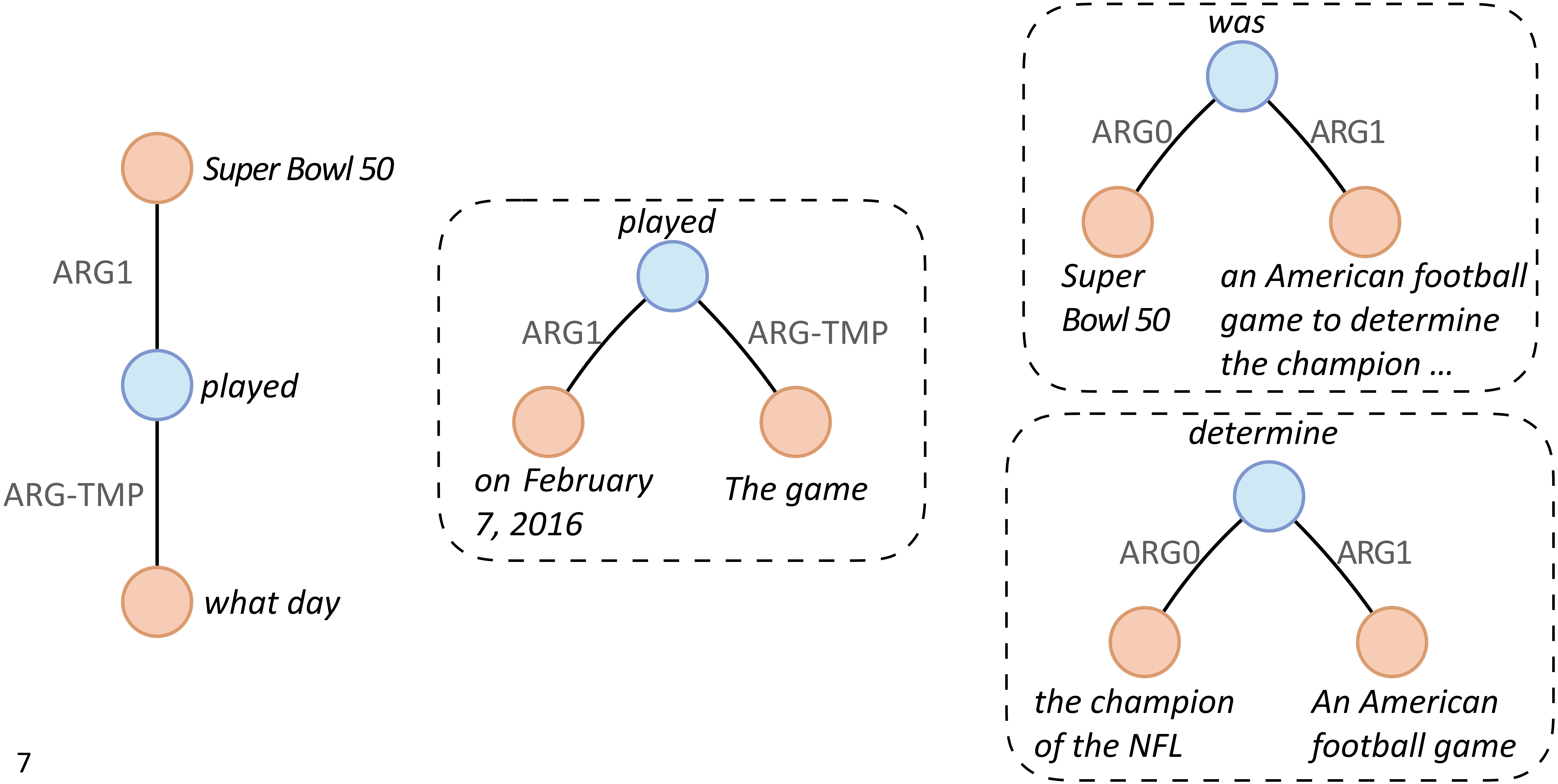
**Question**: What day was Super Bowl 50 played on?



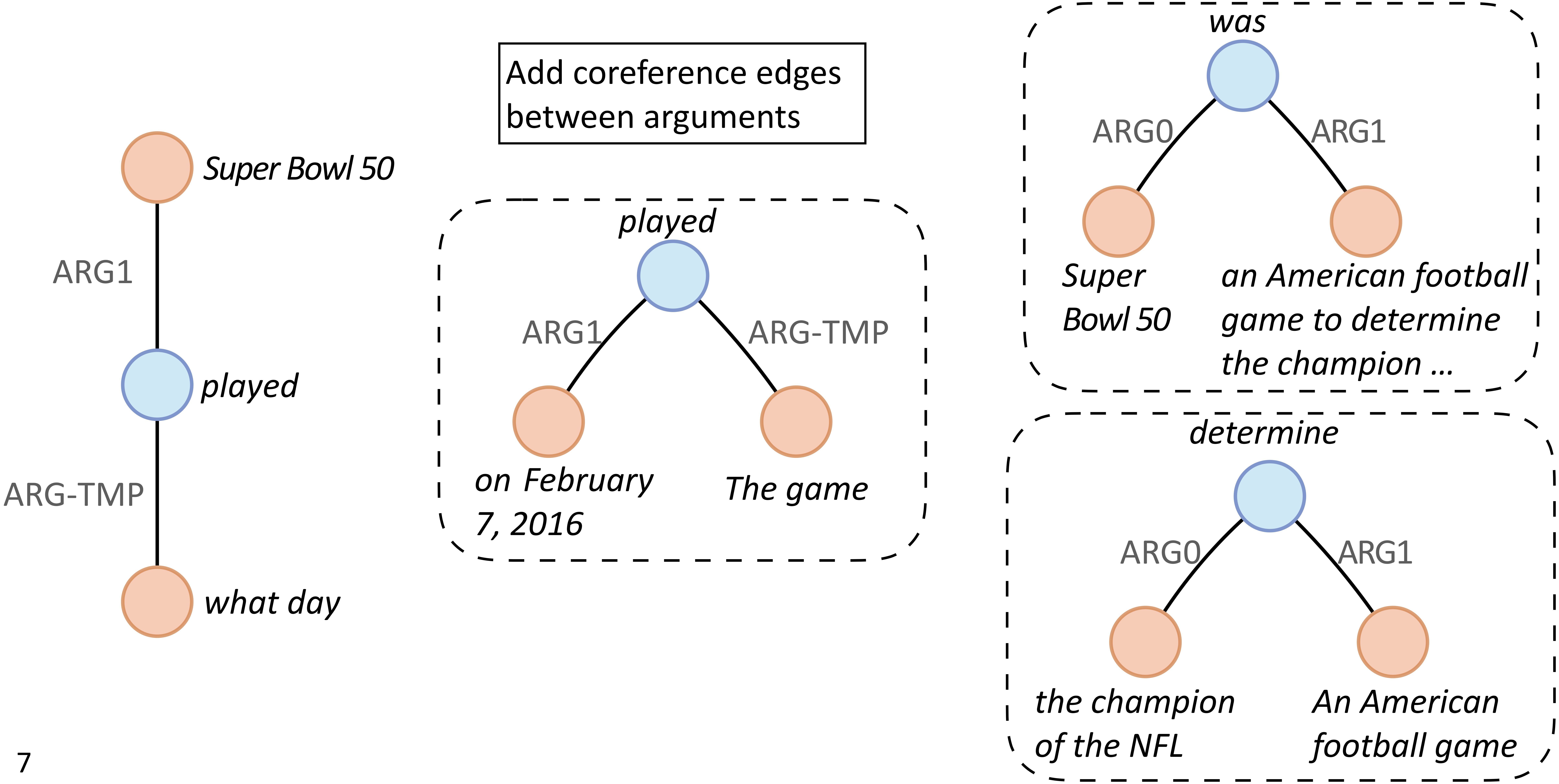

**Context:** Super Bowl 50 was an American football game to determine the champion NFL … The game was played on February 7, 2016



# Graph construction

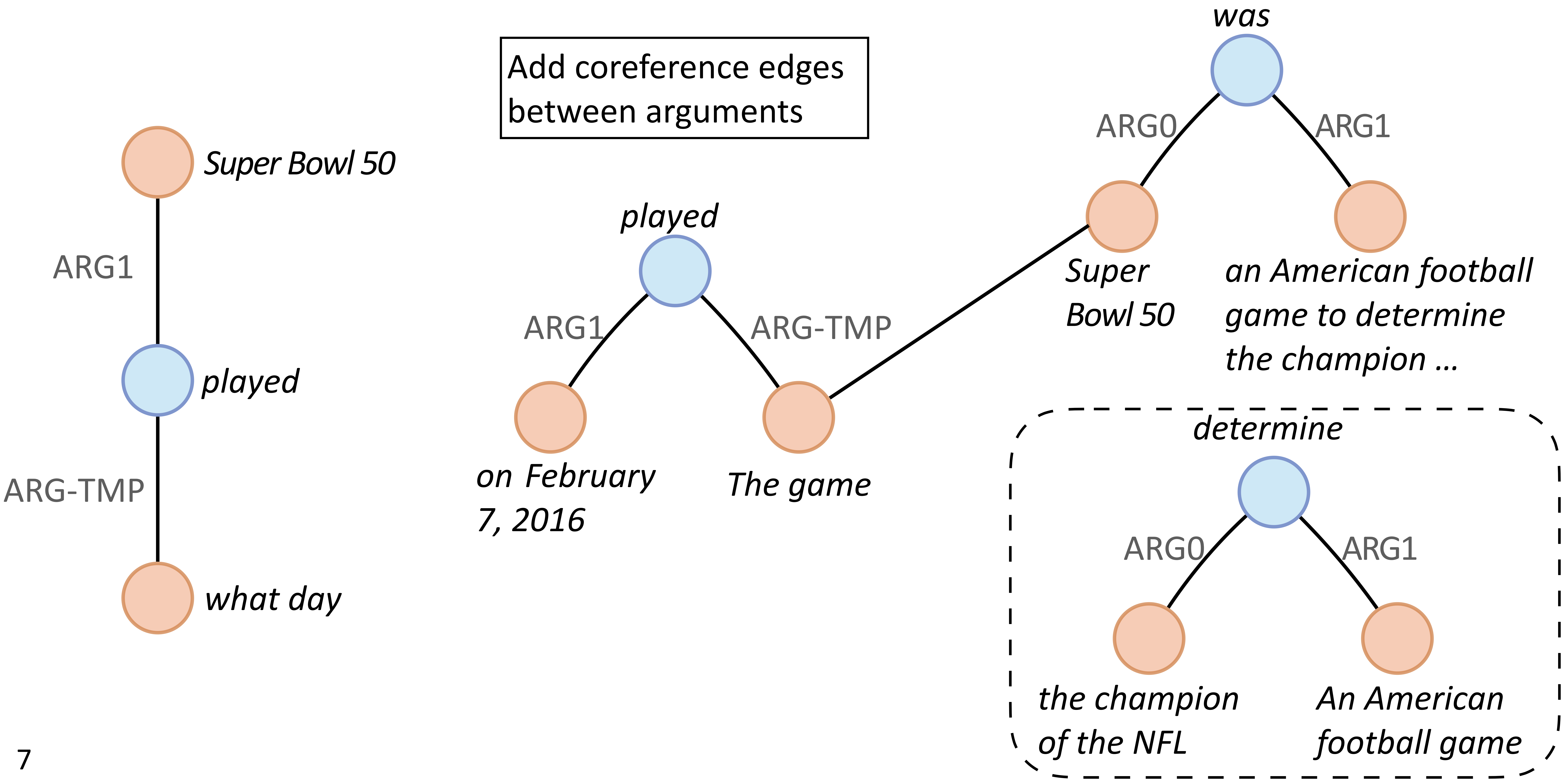


# Graph construction



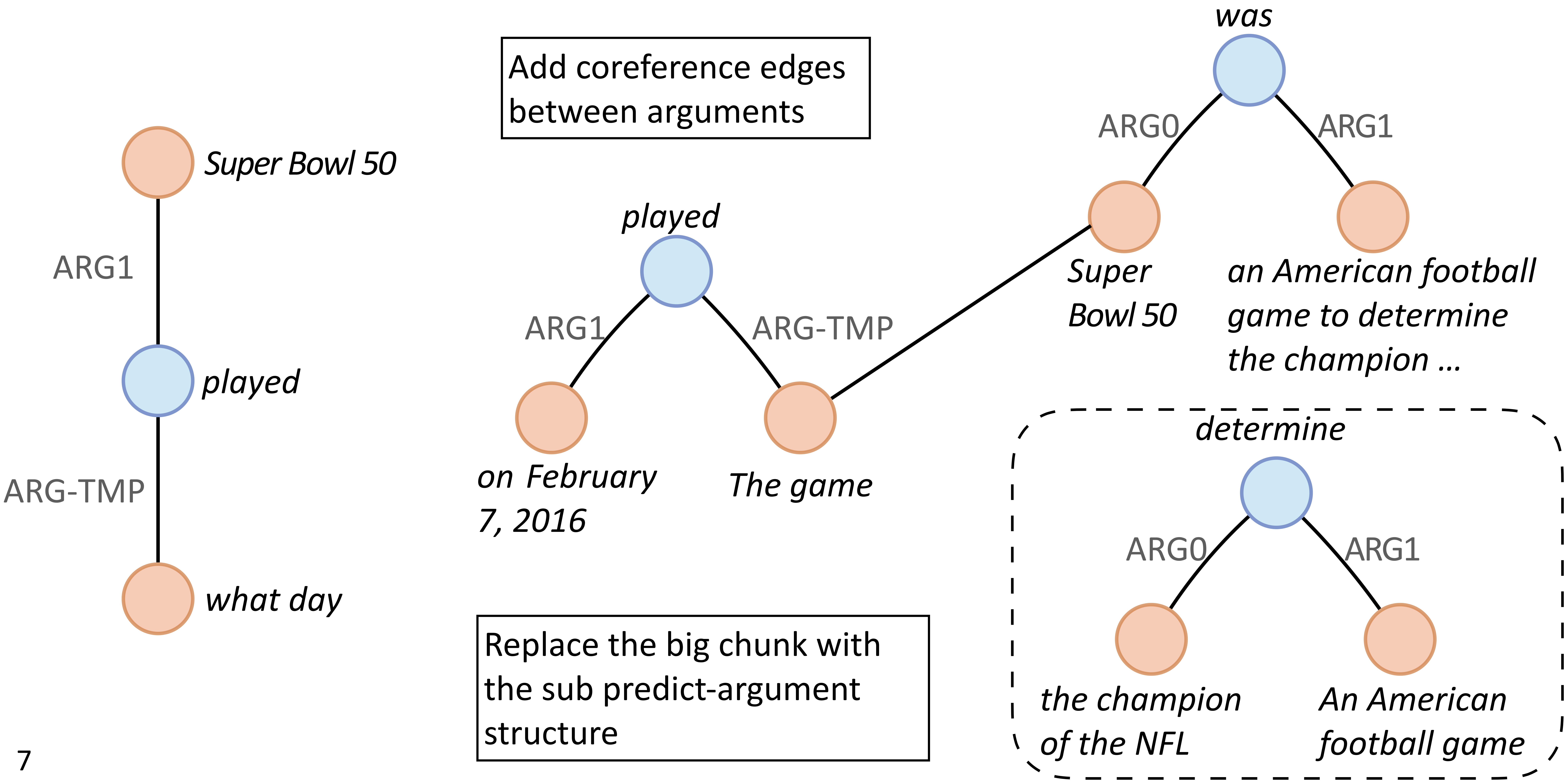
# Graph construction

Add coreference edges between arguments



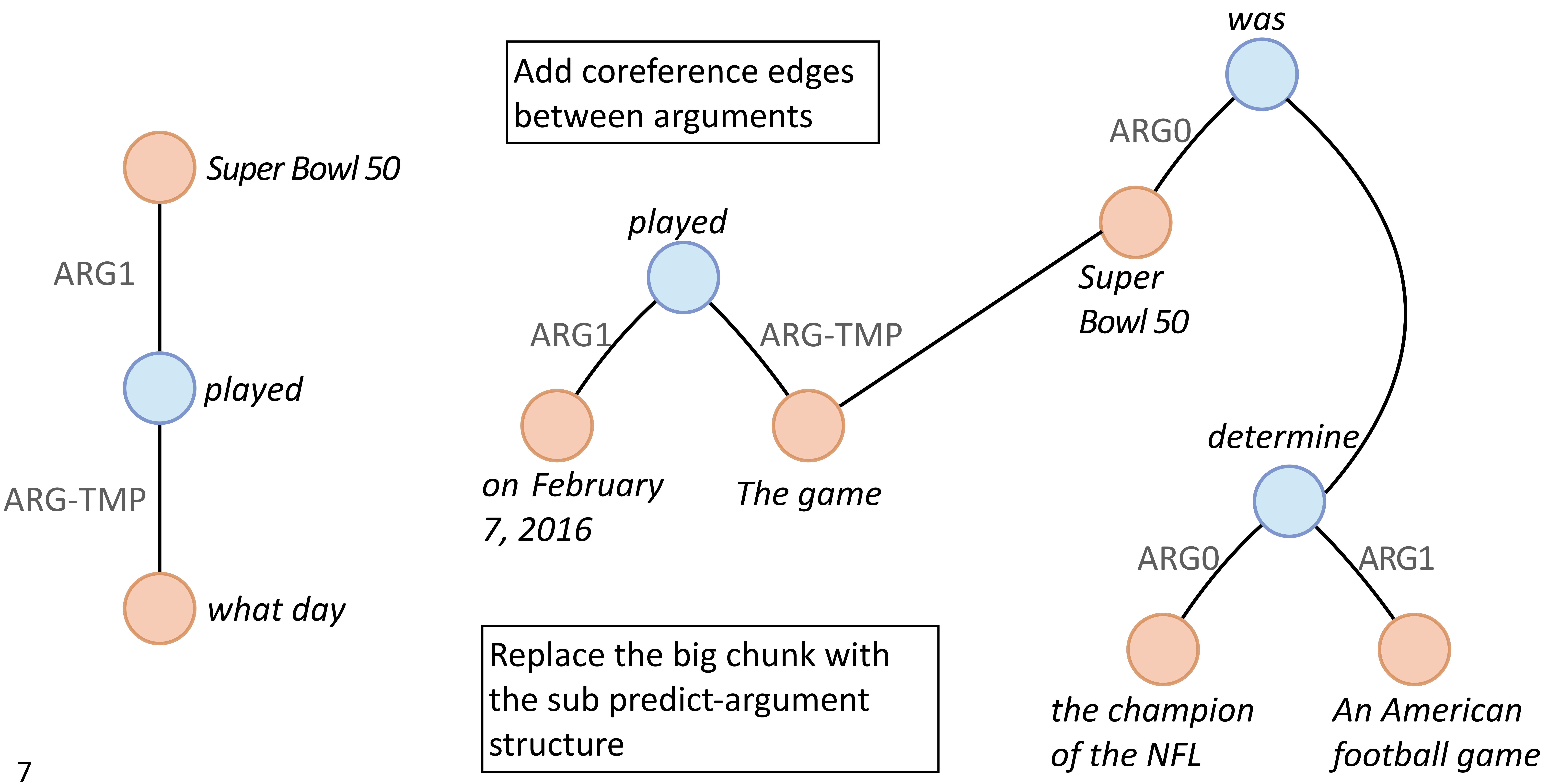
# Graph construction

Add coreference edges between arguments



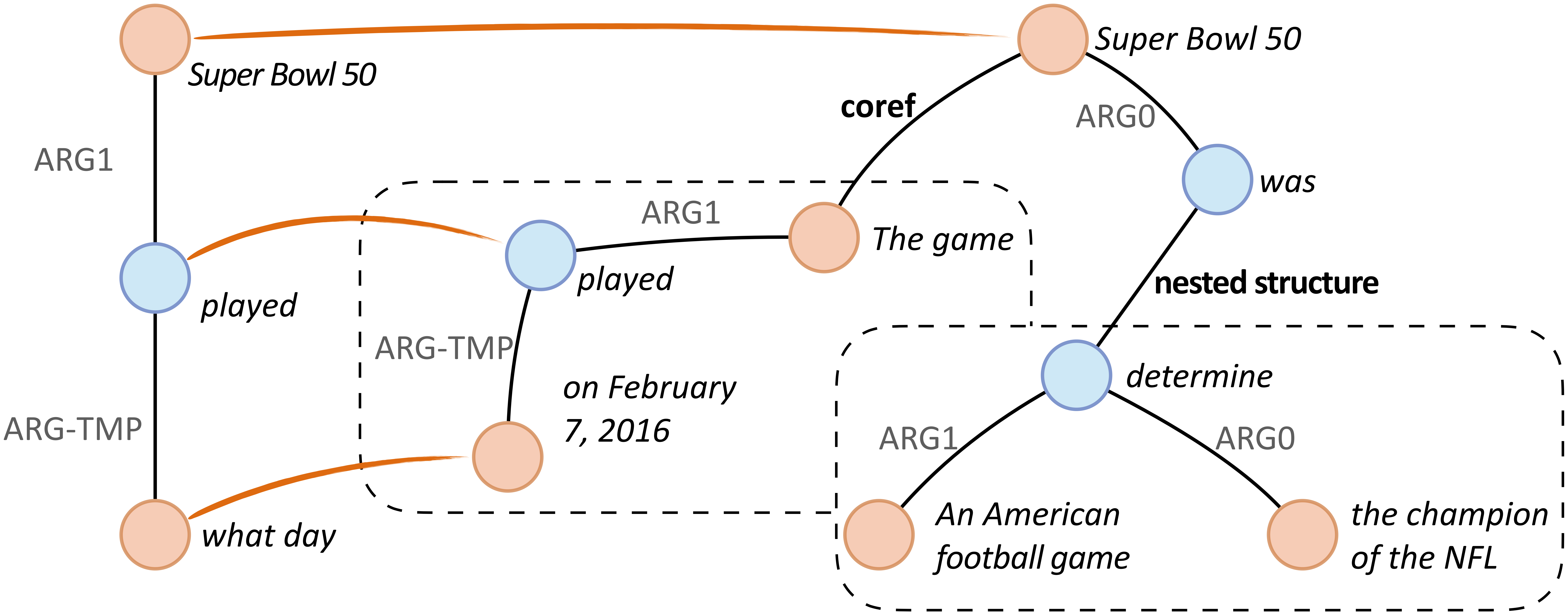

Replace the big chunk with the sub predict-argument structure

# Graph construction



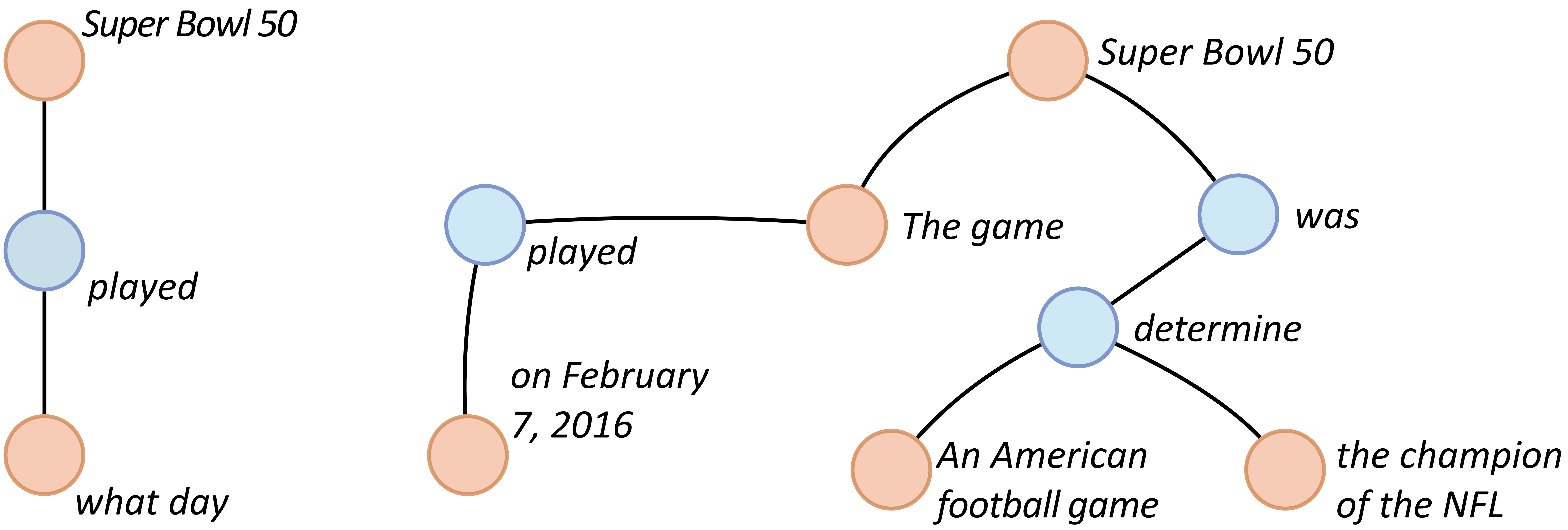
# Graph construction

Find the graph alignment between the question and the context:

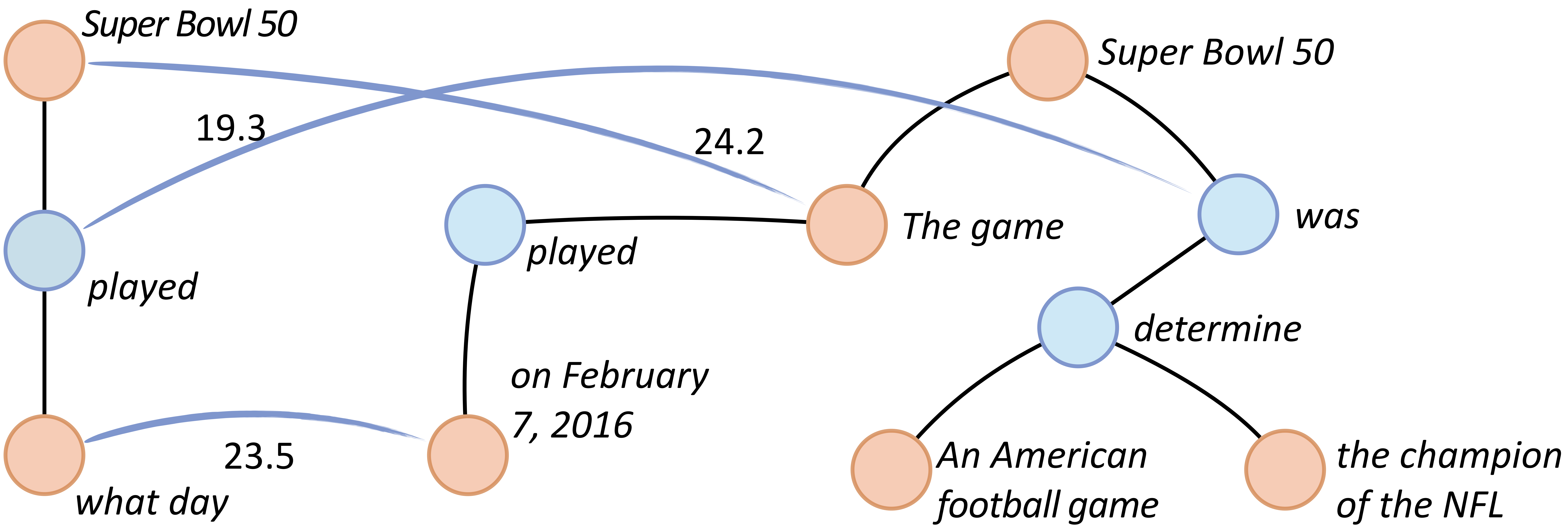


# Model: Find the best graph alignment

# Model: Find the best graph alignment

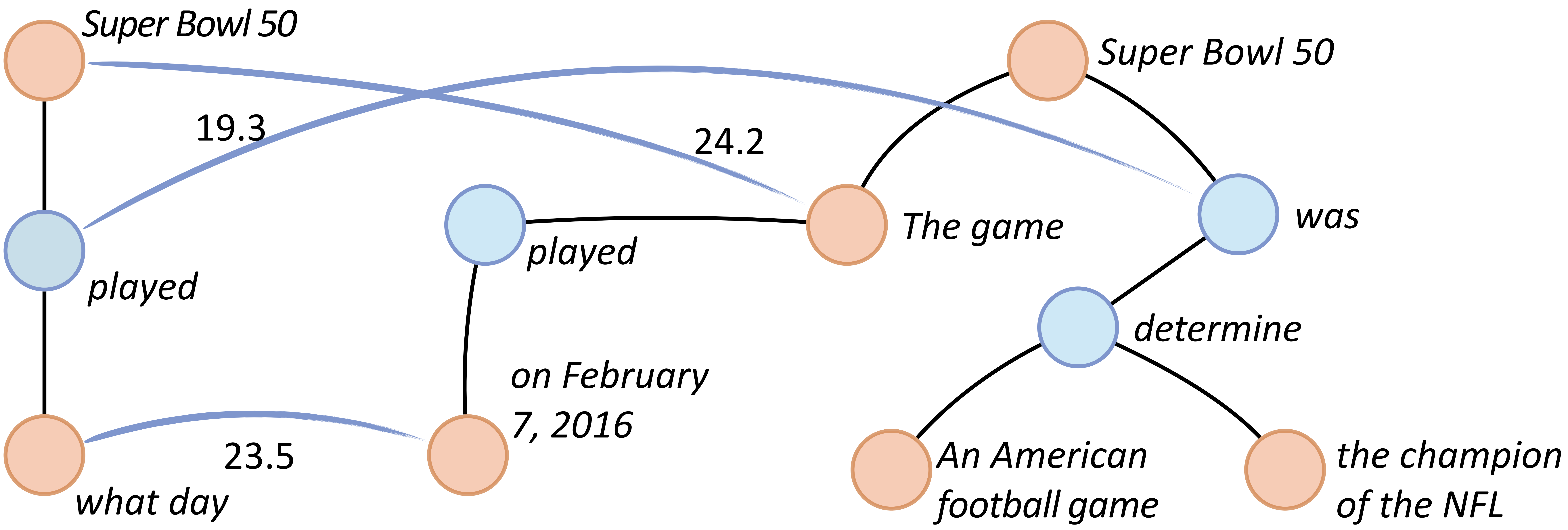


# Model: Find the best graph alignment



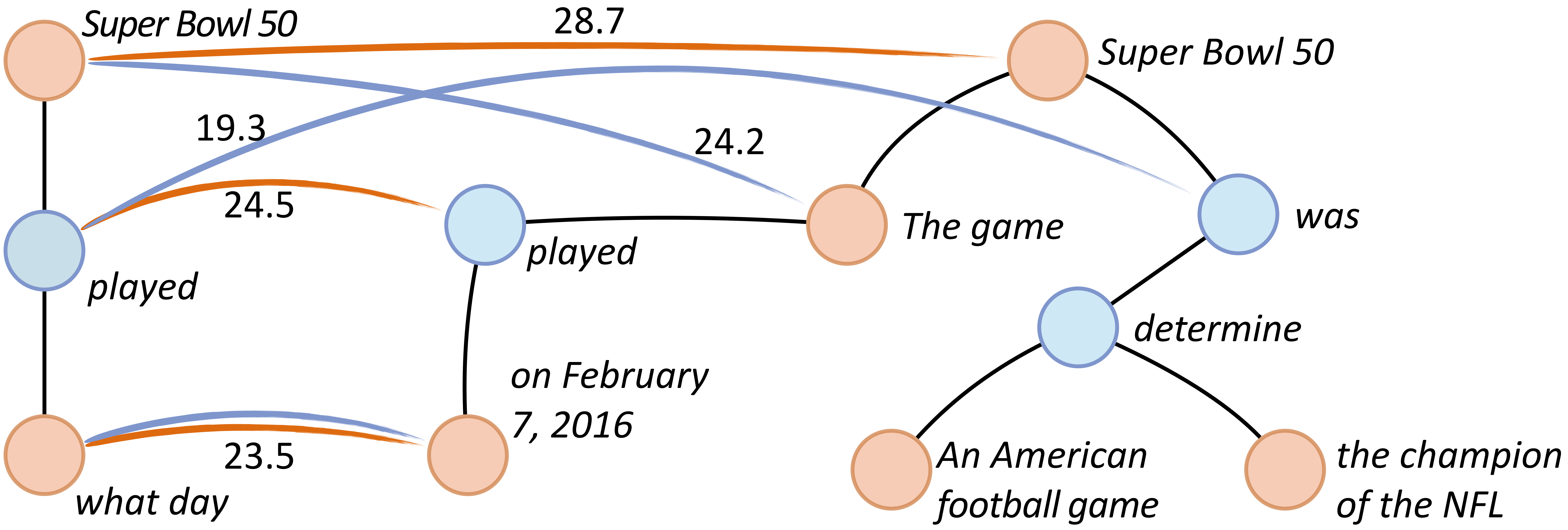
# Model: Find the best graph alignment

- The alignment scores are computed by a BERT-based scoring function



# Model: Find the best graph alignment

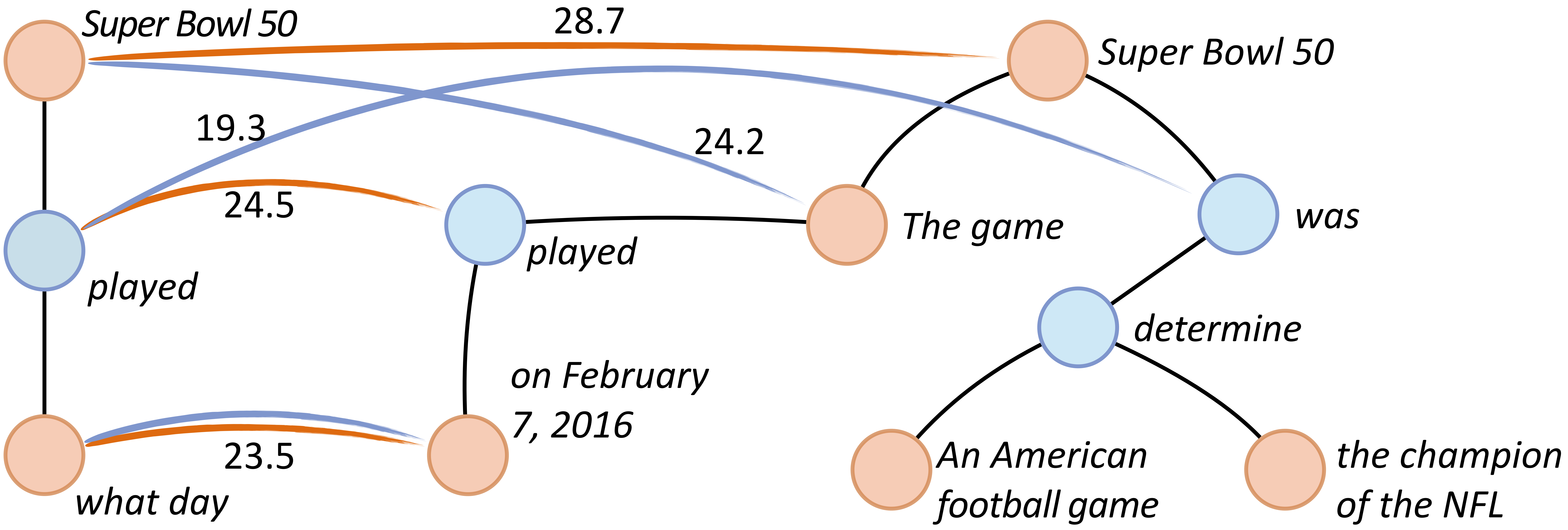
- The alignment scores are computed by a BERT-based scoring function



# Model: Find the best graph alignment

- The alignment scores are computed by a BERT-based scoring function
- Decision is made by sum over all alignment scores

$$23.5 + 24.5 + 28.7 > 23.5 + 19.3 + 24.2$$



# Inference: Beam search w/constraints

# Inference: Beam search w/constraints

- Incrementally build up alignments using beam search subject to constraints

# Inference: Beam search w/constraints

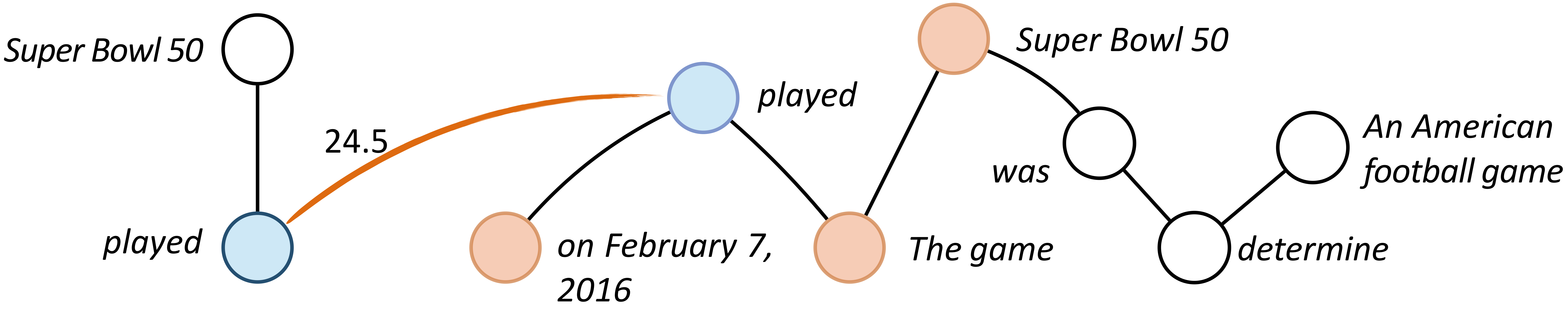
- Incrementally build up alignments using beam search subject to constraints
- Constraints:

  - Locality: adjacent nodes in question should align to nearby nodes in context graph

# Inference: Beam search w/constraints

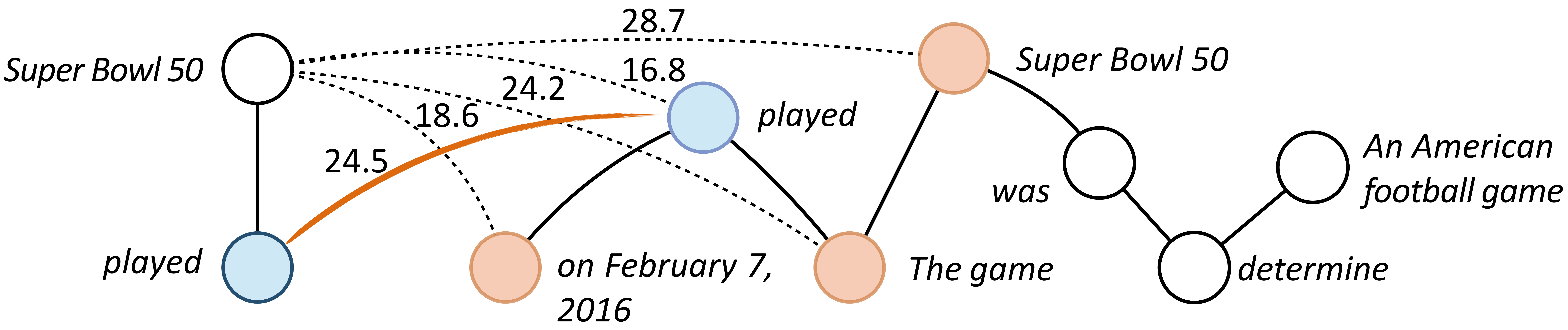
- Incrementally build up alignments using beam search subject to constraints
- Constraints:

  - Locality: adjacent nodes in question should align to nearby nodes in context graph



# Inference: Beam search w/constraints

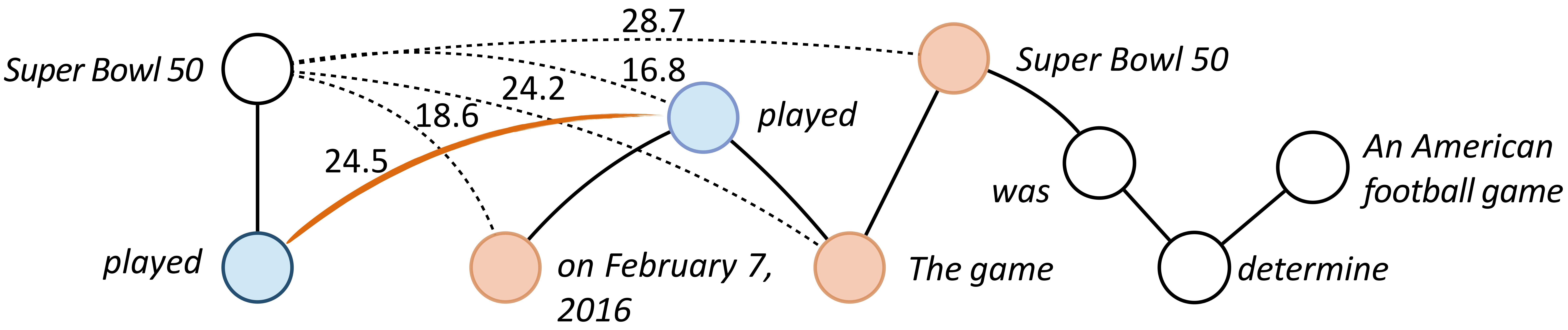
- Incrementally build up alignments using beam search subject to constraints
- Constraints:
  - Locality: adjacent nodes in question should align to nearby nodes in context graph



# Inference: Beam search w/constraints

- Incrementally build up alignments using beam search subject to constraints
- Constraints:

  - Locality: adjacent nodes in question should align to nearby nodes in context graph

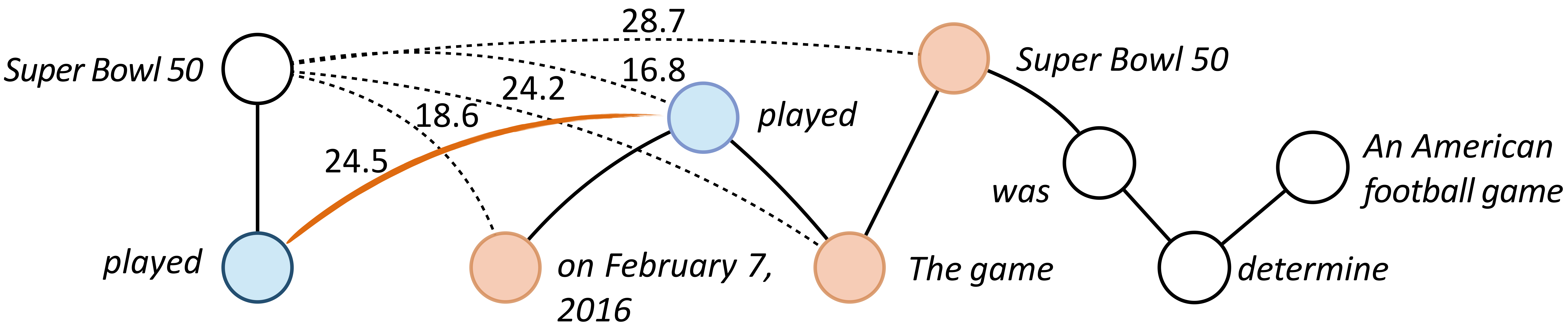



  - Entity constraint (later in the talk): require hard entity match

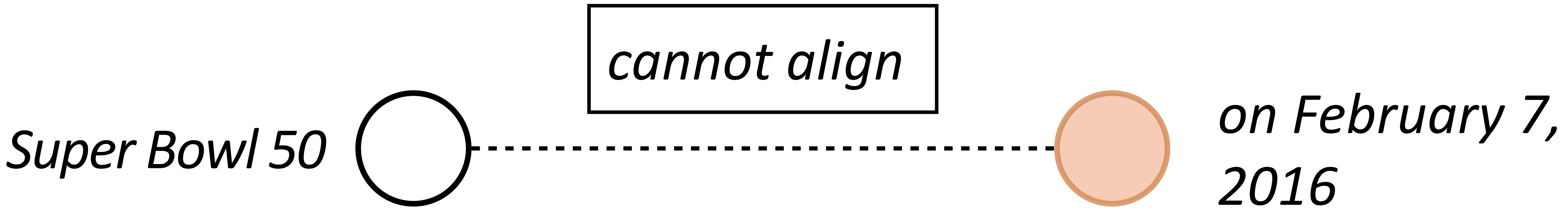
# Inference: Beam search w/constraints

- Incrementally build up alignments using beam search subject to constraints
- Constraints:

  \- Locality: adjacent nodes in question should align to nearby nodes in context graph

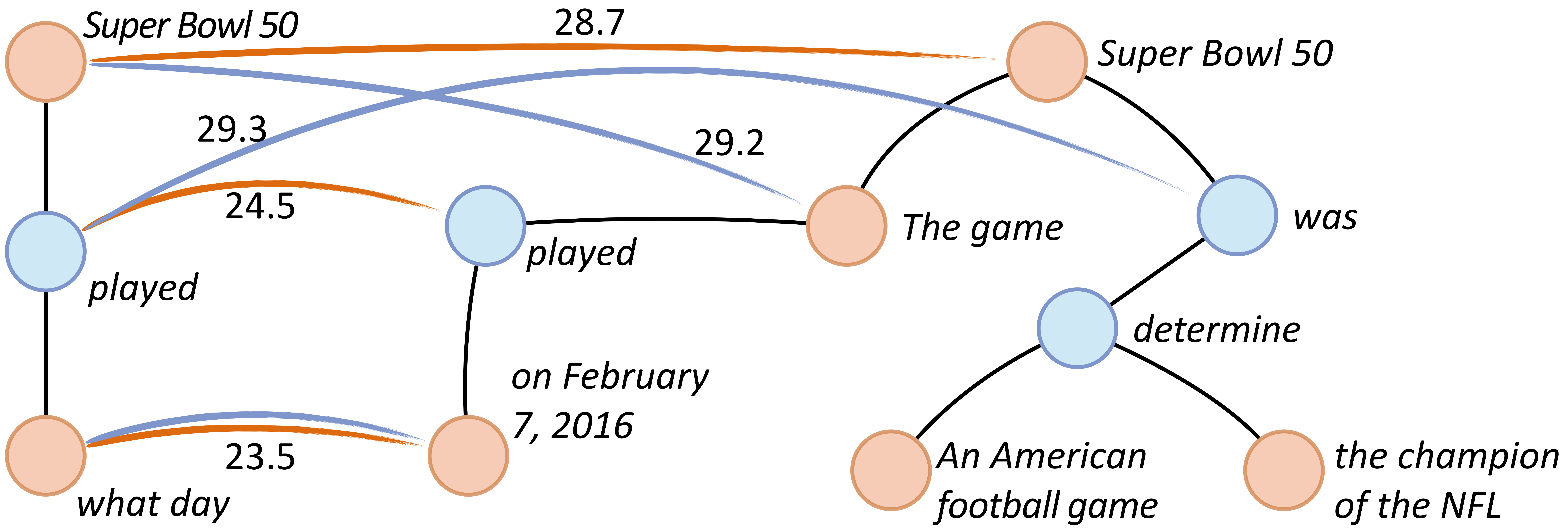



  \- Entity constraint (later in the talk): require hard entity match



# Global Training: SSVM w/beam search

- Global training:
  - Decision is made by sum over all alignment scores

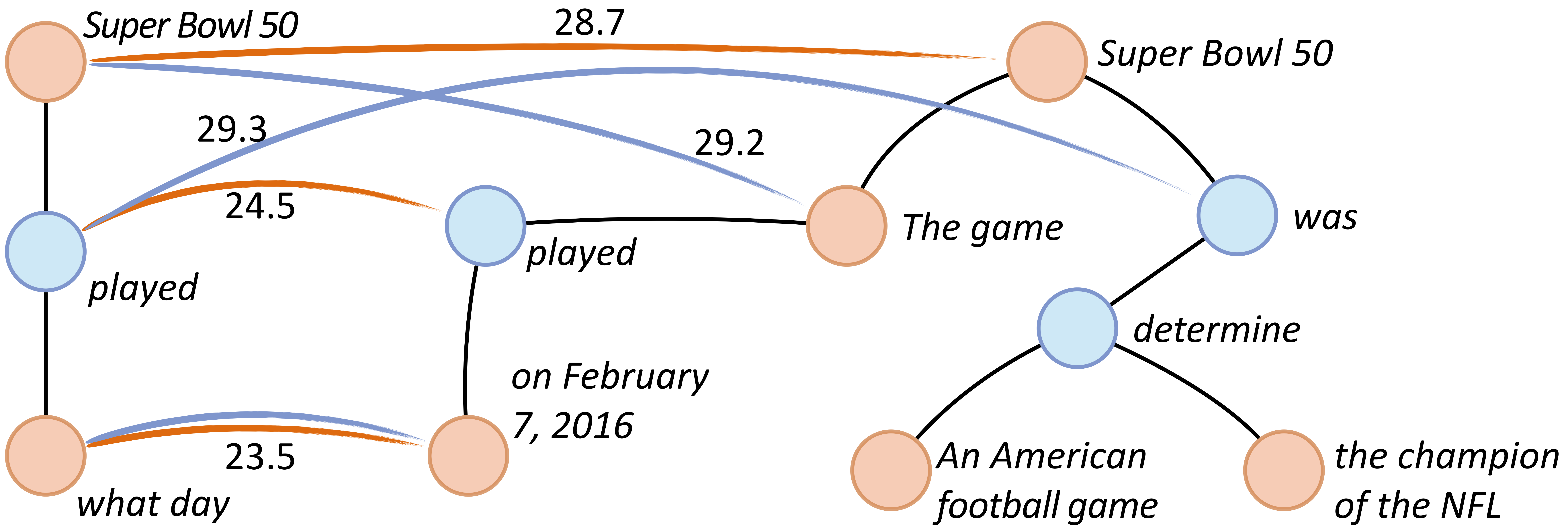


# Global Training: SSVM w/beam search

- Global training:
  - Decision is made by sum over all alignment scores

$$\mathcal{L} = \max(0, \max_{\mathbf{a} \in \mathcal{A}} [f(\mathbf{a}, \mathbf{Q}, \mathbf{C}) + \mathrm{Ham}(\mathbf{a}^*, \mathbf{a})] - f(\mathbf{a}^*, \mathbf{Q}, \mathbf{C}))$$ Loss of SSVM

(29.3 + 29.2 + 23.5) + 2 - (23.5 + 24.5 + 28.7) = 7.2



# Outline

1) Question answering via sub-part alignment
   - Graph construction
   - Model: **graph alignment** between the question and the context
   - Inference: beam search respecting **constraints**
   - Training: **SSVM** using beam search

2) Experiments
   - Adversarial robustness
   - Constraints on alignment scores

3) Takeaways

# Dataset

# Dataset

Training:

- SQuAD-1.1 — Standard benchmark

# Dataset

Training:

- SQuAD-1.1 — Standard benchmark

Testing:

- SQuAD-adversarial — append human approved strong distractors to the original context

  - Two datasets, SQuAD-addSent and SQuAD-addOneSent

**Context**: Super Bowl 50 was an American football game to determine the champion of NFL … The game was played on February 7, 2016 …

Append

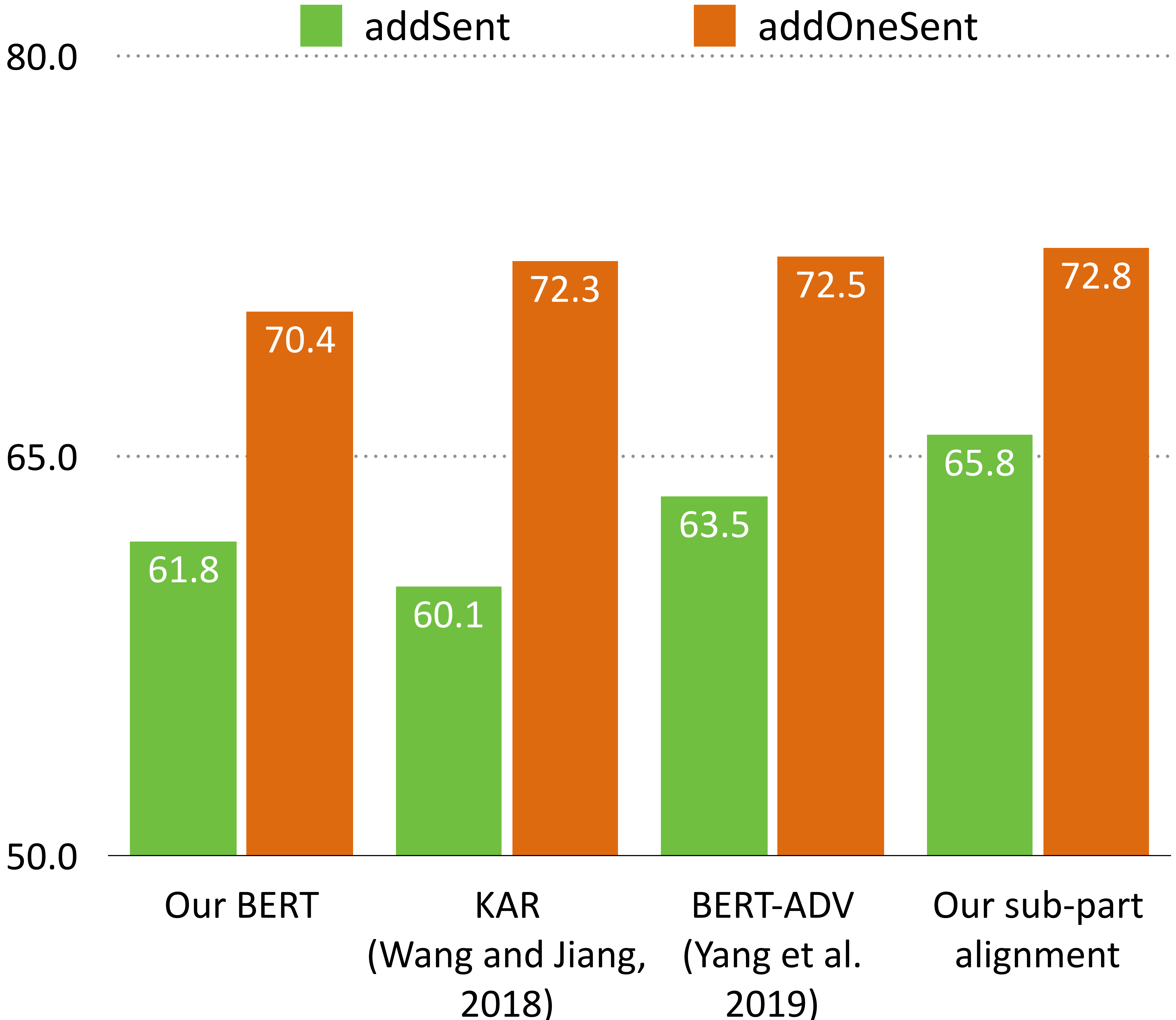
**Adversarial Context**: The Champ Bowl was played on the day of August 18, 1991

# Adversarial robustness

Systems:

KAR: Explicit knowledge injection

BERT-ADV: Adversarial training on BERT



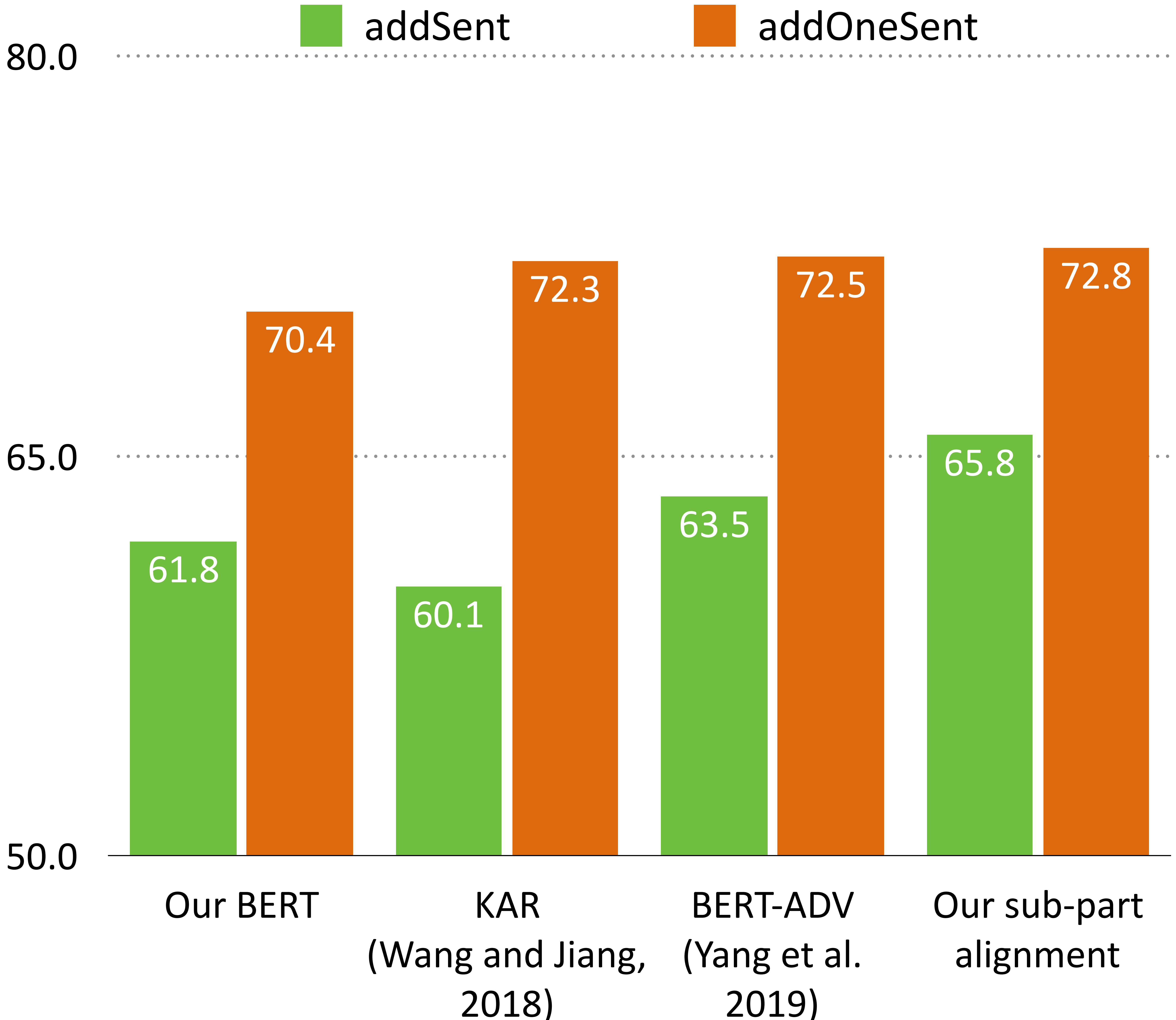
# Adversarial robustness

Systems:

KAR: Explicit knowledge injection

BERT-ADV: Adversarial training on BERT

- Our sub-part alignment system largely outperforms the BERT baseline and several systems in the literature.



# Constraint on entities

# Constraint on entities

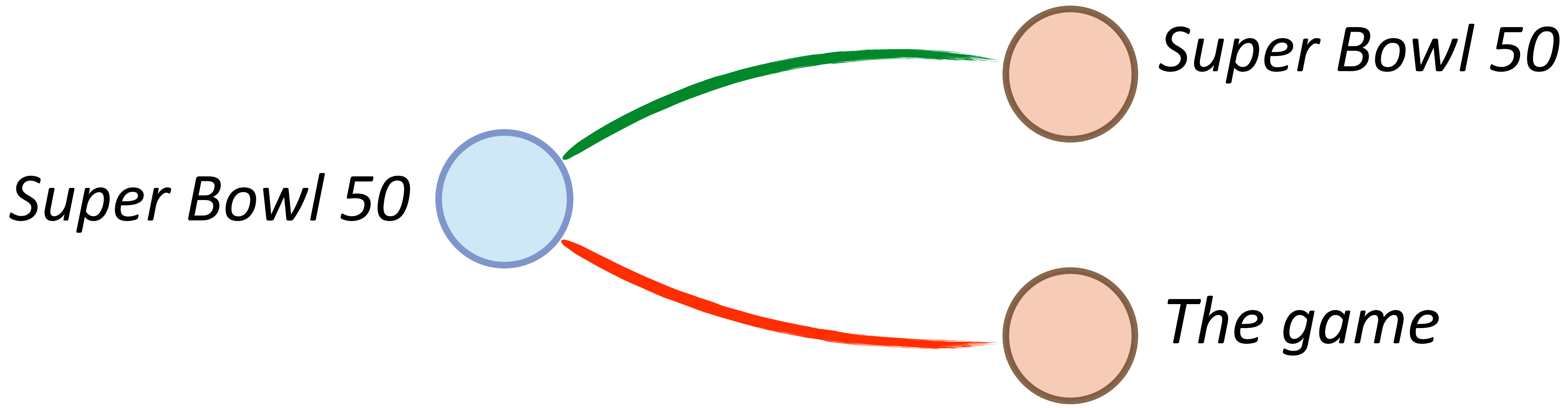
- Explicit alignments allow us to control the model's behavior
  - Reject unreliable predictions to trade coverage for performance — If the model could choose to answer k percentage of examples, how well does it do? (Selective QA setting, Kamath et al. 2020)

# Constraint on entities

- Explicit alignments allow us to control the model's behavior
  - Reject unreliable predictions to trade coverage for performance — If the model could choose to answer k percentage of examples, how well does it do? (Selective QA setting, Kamath et al. 2020)

- Constraint on entity matches:

  Force hard entity match

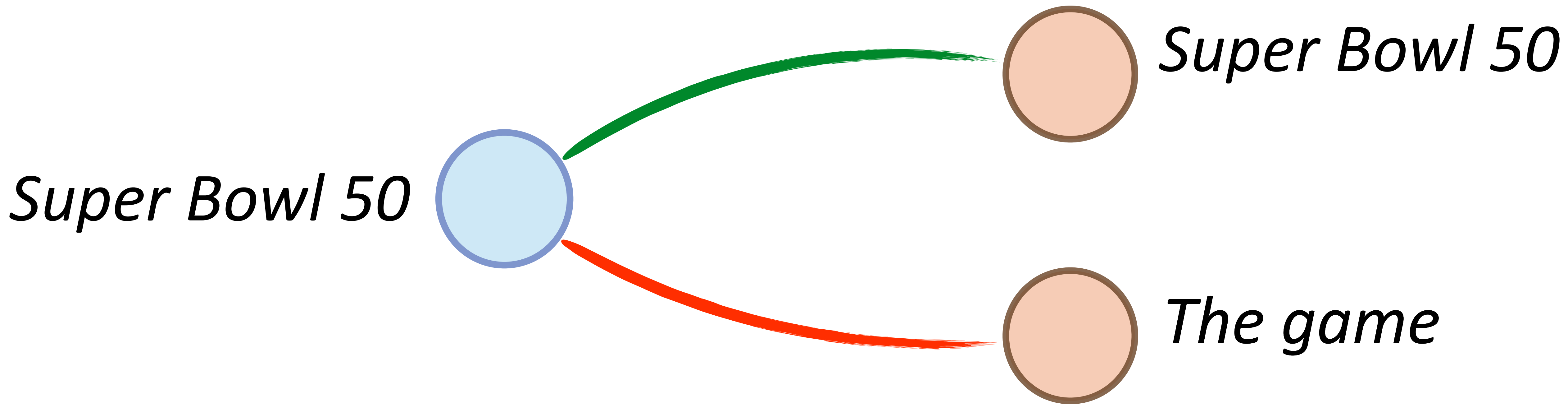


# Constraint on entities

- Explicit alignments allow us to control the model's behavior
  - Reject unreliable predictions to trade coverage for performance — If the model could choose to answer k percentage of examples, how well does it do? (Selective QA setting, Kamath et al. 2020)

- Constraint on entity matches:

  Force hard entity match




- Throw out the examples without a hard entity match

# Constraint on alignment scores

# Constraint on alignment scores

- Constraint on alignment scores:

  - Alignment scores we produced are good indicators of how well the alignments are

# Constraint on alignment scores

- Constraint on alignment scores:

  - Alignment scores we produced are good indicators of how well the alignments are

**Question**: Who created an engine using high pressure steam in 1801?

30.1 33.1 26.3 27.3 28.3 18.8

**Adversarial alignment**: Jeff Dean created an engine using low pressure steam in 1790.

# Constraint on alignment scores

- Constraint on alignment scores:

  - Alignment scores we produced are good indicators of how well the alignments are

**Question**: Who created an engine using high pressure steam in 1801?

30.1 33.1 26.3 27.3 28.3 18.8

**Adversarial alignment**: Jeff Dean created an engine using low pressure steam in 1790.

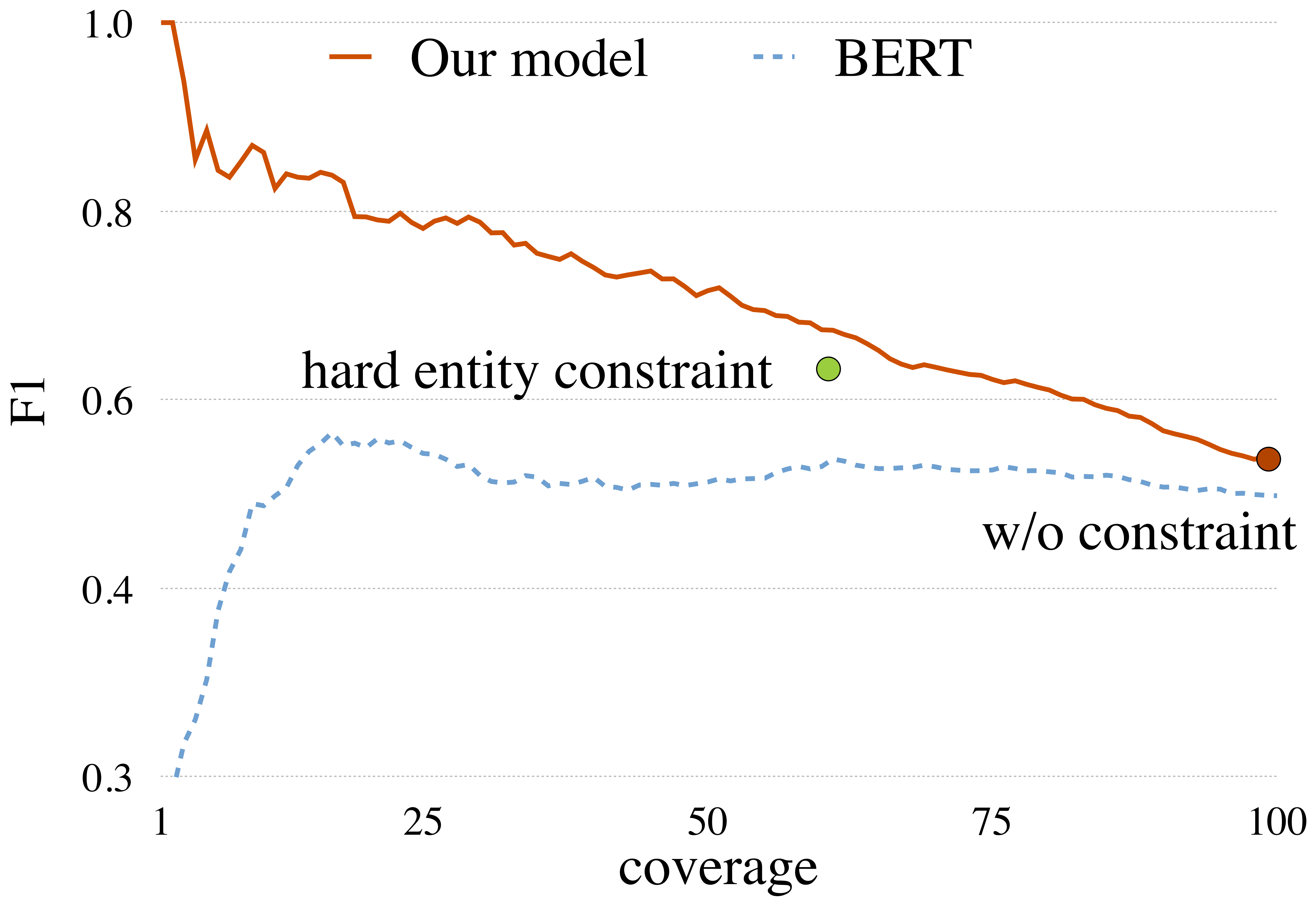
- How to find the unreliable alignment:

  - Worst Link Gap: max score over all alignments - min score over the prediction

  - Larger Worst Link Gap indicates lower confidence in prediction

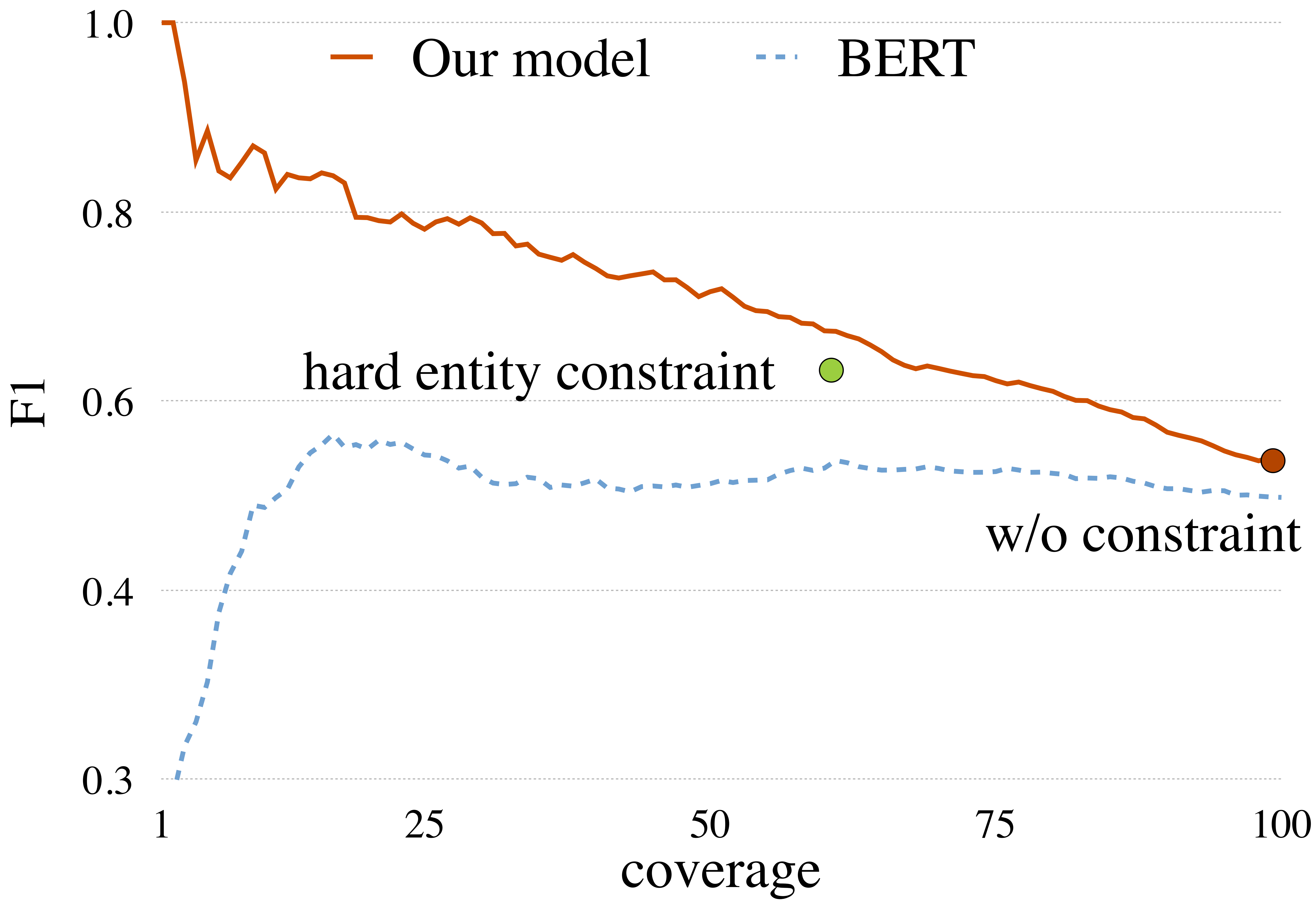
# Constrained performance

- If our model can choose to answer only the k percentage of examples it's most confident about (the coverage), what F1 does it achieve?



# Constrained performance

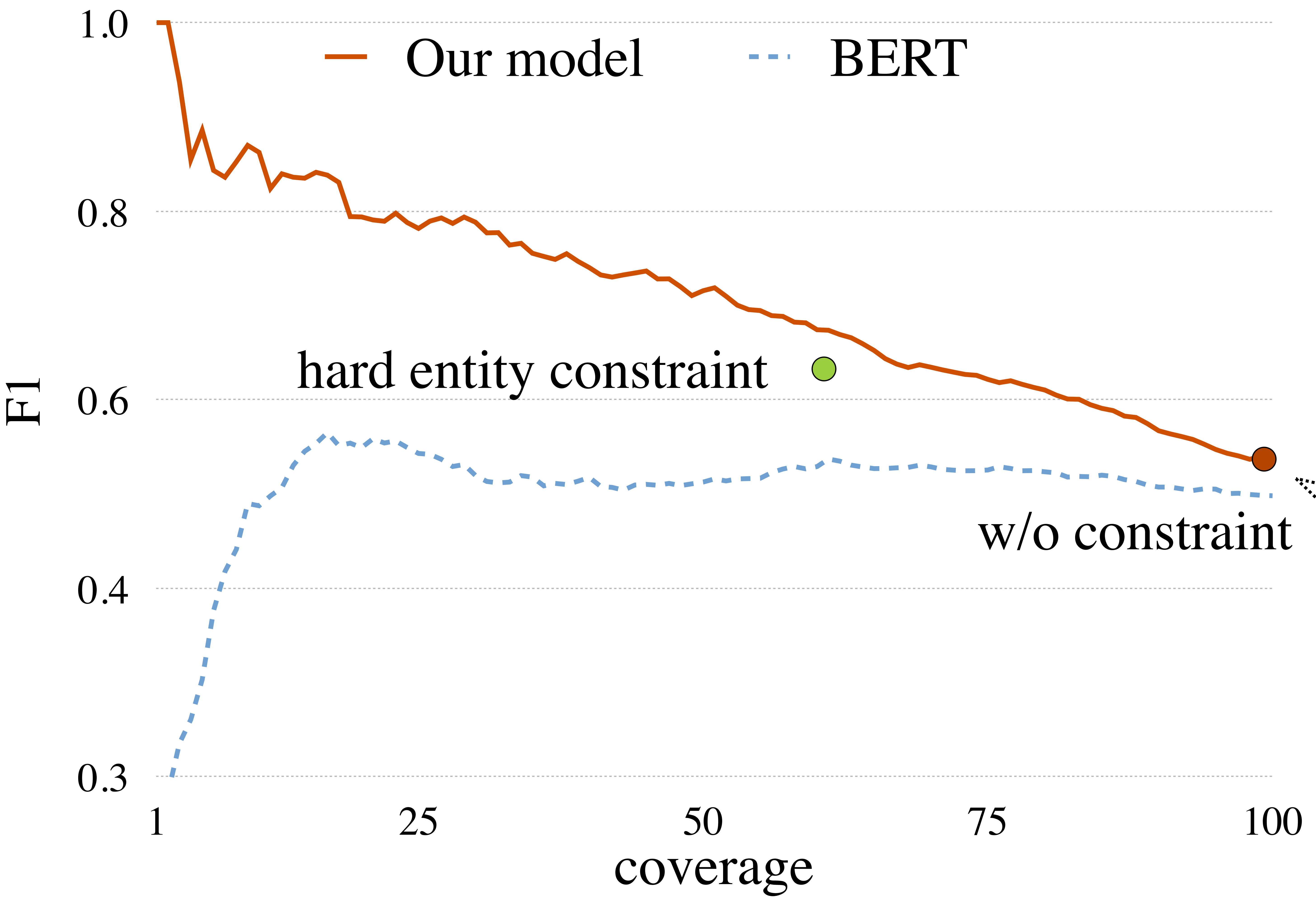
- If our model can choose to answer only the k percentage of examples it's most confident about (the coverage), what F1 does it achieve?




- For our model, the confidence is taken to be the Worst Link Gap; For BERT, the confidence is posterior probability.

# Constrained performance

- If our model can choose to answer only the k percentage of examples it's most confident about (the coverage), what F1 does it achieve?

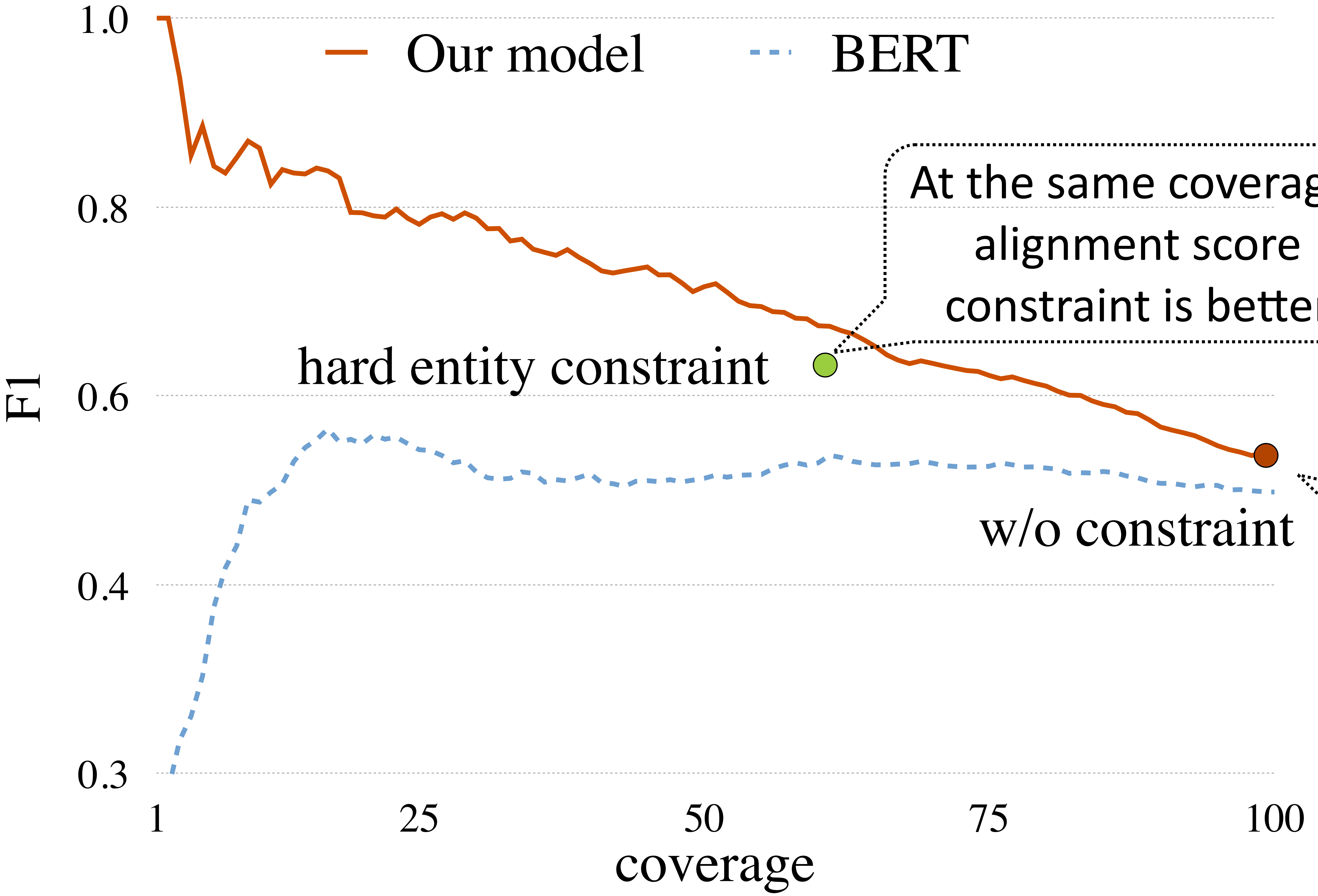



- For our model, the confidence is taken to be the Worst Link Gap; For BERT, the confidence is posterior probability.

The confidence scores of BERT QA do not align with its performance, while our alignment score is well calibrated

# Constrained performance

- If our model can choose to answer only the k percentage of examples it's most confident about (the coverage), what F1 does it achieve?




At the same coverage, alignment score constraint is better

The confidence scores of BERT QA do not align with its performance, while our alignment score is well calibrated

- For our model, the confidence is taken to be the Worst Link Gap; For BERT, the confidence is posterior probability.

# Outline

1) Question answering via sub-part alignment
   - Graph construction
   - Model: **graph alignment** between the question and the context
   - Inference: beam search respecting **constraints**
   - Training: **SSVM** using beam search

2) Experiments
   - Adversarial robustness
   - Constraints on alignment scores

3) Takeaways

# Takeaways

# Takeaways

- The subpart-alignment is a viable way of verifying whether the whole question is supported by the context.
  - It makes the QA process more explicit, thus more explainable and debuggable
  - It allows us to place explicit constraints to gain more control of the model

# Takeaways

- The subpart-alignment is a viable way of verifying whether the whole question is supported by the context.
  - It makes the QA process more explicit, thus more explainable and debuggable
  - It allows us to place explicit constraints to gain more control of the model
- Identifying the misalignment between the question and the context is hard
  - How to automatically identify and align the spans — SRL is inflexible and doesn't cover everything
  - Noun phrase alignment is easy to learn while the predicate alignment is hard
  - Check our new preprint on using an entailment model to aid the alignment process

Thank you!