Feature-Less End-to-End Nested Term Extraction

Yuze Gao and Yu Yuan

Oct. 12, 2019, Dunhuang

Outline

- Background and Methods
- Our Model
- Experiments and Results
- Conclusion

Outline

- Background and Methods
- Our Model
- Experiments and Results
- Conclusion

Background and Methods

- Automatic Term Extraction (ATE)
  - Give a sequence s_n, find and extract domain specified phrases

Background and Methods

- Automatic Term Extraction (ATE)
  - Give a sequence s_n, find and extract domain specified phrases

S_n :

0	1	2	3	4	5
Mouse	interleukin-2	receptor	alpha	gene	expression

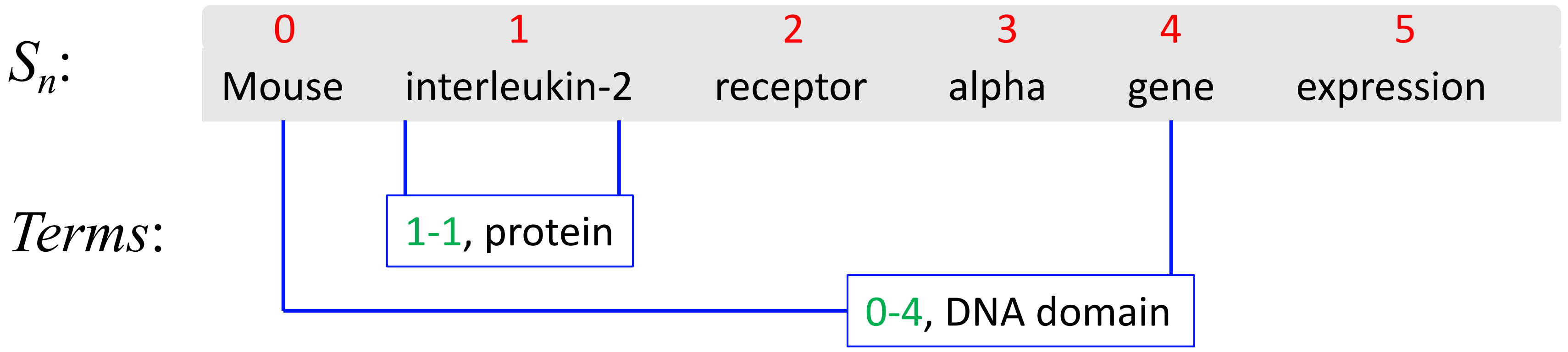
Background and Methods

- Automatic Term Extraction (ATE)
  - Give a sequence s_n, find and extract domain specified phrases

	0	1	2	3	4	5
S_n:	Mouse	interleukin-2	receptor	alpha	gene	expression
Terms:		1-1, protein				

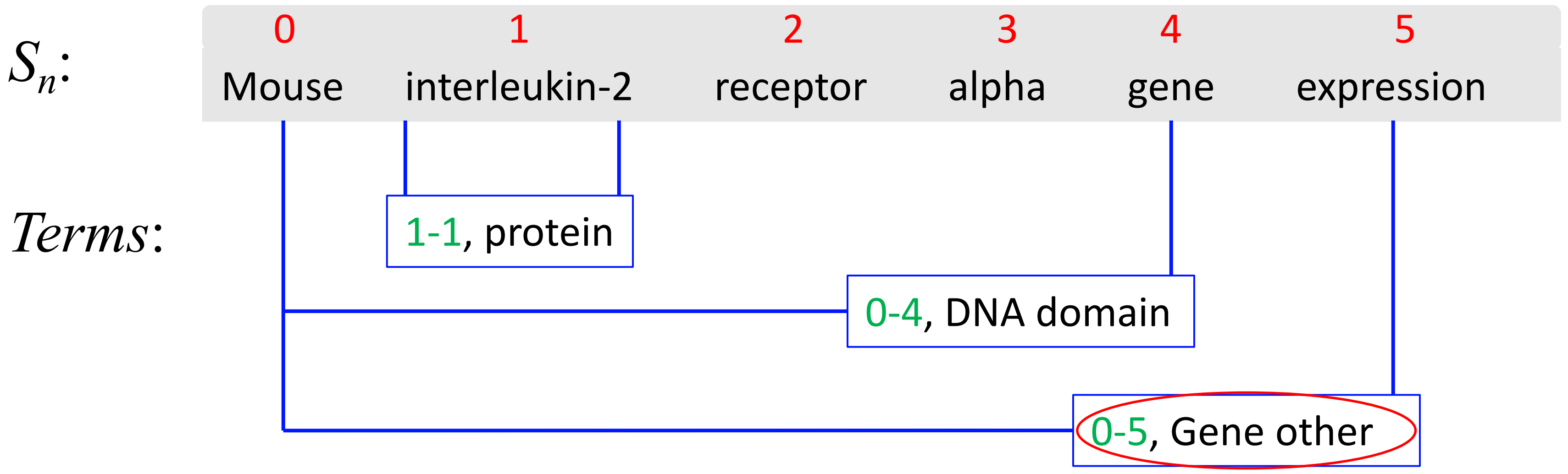
Background and Methods

- Automatic Term Extraction (ATE)
  - Give a sequence s_n, find and extract domain specified phrases



Background and Methods

- Automatic Term Extraction (ATE)
  - Give a sequence s_n, find and extract domain specified phrases



Background and Methods

- Automatic Term Extraction (ATE)
- Methods:
    - Feature-based: Yu et al. 2017

Yu Yuan, Jie Gao and Yue Zhang: "Supervised learning for robust term extraction." 2017 International Conference on Asian Language Processing (IALP). IEEE, 2017.

Background and Methods

- Automatic Term Extraction (ATE)
- Methods:
  - Feature-based: Yu et al. 2017:
    - Using ten different extracted features (such as IDF, consine distance and etc.)

Yu Yuan, Jie Gao and Yue Zhang: "Supervised learning for robust term extraction." 2017 International Conference on Asian Language Processing (IALP). IEEE, 2017.

Background and Methods

- Automatic Term Extraction (ATE)
- Methods:
  - Feature-based: Yu et al. 2017:
    - Using ten different extracted features (such as IDF, consine distance and etc.)
    - Drawback: Time Consuming and complicate in feature preparation

Yu Yuan, Jie Gao and Yue Zhang: “Supervised learning for robust term extraction.” 2017 International Conference on Asian Language Processing (IALP). IEEE, 2017.

Background and Methods

- Automatic Term Extraction (ATE)
- Methods:
  - Feature-based: Yu et al. 2017

Yu Yuan, Jie Gao and Yue Zhang: "Supervised learning for robust term extraction." 2017 International Conference on Asian Language Processing (IALP). IEEE, 2017.

Background and Methods

- Automatic Term Extraction (ATE)
- Methods:
    - Feature-based: Yu et al. 2017
    - DNN Sequence Labelling-based: Basaldella et al. 2018:

Basaldella M, Antolli E, Serra G: ”Bidirectional lstm recurrent neural network for key phrase extraction”, Italian Research Conference on Digital Libraries. Springer, Cham, 2018: 180-187.

Background and Methods

- Automatic Term Extraction (ATE)
- Methods:
  - Feature-based: Yu et al. 2017
  - DNN Sequence Labelling-based: Basaldella et al. 2018:
    - Using Sequence Labelling method on LSTM NN module

Basaldella M, Antolli E, Serra G: "Bidirectional lstm recurrent neural network for key phrase extraction", Italian Research Conference on Digital Libraries. Springer, Cham, 2018: 180-187.

Background and Methods

- Automatic Term Extraction (ATE)
- Methods:
  - Feature-based: Yu et al. 2017
  - DNN Sequence Labelling-based: Basaldella et al. 2018:
    - Using Sequence Labelling method on LSTM NN module
    - Drawback: does not support nested term

Basaldella M, Antolli E, Serra G: ”Bidirectional lstm recurrent neural network for key phrase extraction”, Italian Research Conference on Digital Libraries. Springer, Cham, 2018: 180-187.

Background and Methods

- Problems:
  - Nested Term is very common in terminology extraction.
  - Feature-based methods call for prepared features and the preparation is time-consuming and complex.
  - Sequence Labelling Methods do not support nested term extraction.
  - Most existing systems do not take advantage of information from sentence level

Outline

- Background and Methods
- Our Model
- Experiments and Results
- Conclusion

Our Model

- Span-based Term Extraction
  - Treat every span within a fixed length as a potential term

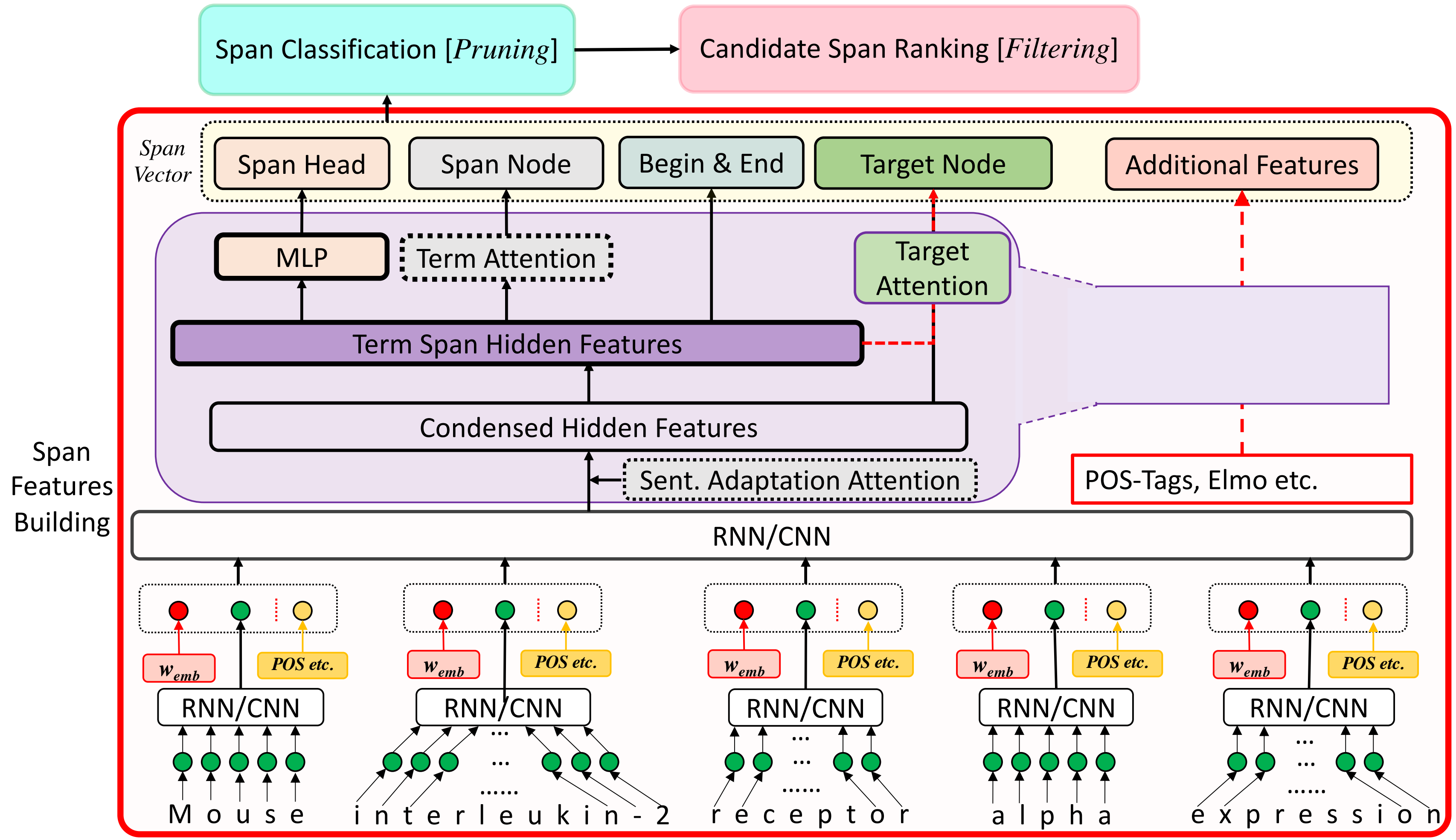
Sentence ($\boldsymbol{n}$=6)	"Mouse interleukin-2 receptor alpha gene expression"
True Term Spans	[0, 4], [0, 5], [1, 1]
Model Processed Spans ($\boldsymbol{k}$=5)	[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [1, 1], [1, 2], [1, 3], [1, 4], [1, 5], [2, 2], [2, 3], [2, 4], [2, 5], [3, 3], [3, 4], [3, 5], [4, 4], [4, 5], [5, 5]

Our Model

- Span-based Term Extraction
  - Treat every span within fixed length as a potential term.
  - The span is used as processed unit, represent every span with a vector.
  - External feature is not a must, we build feature patterns internally from hidden output.

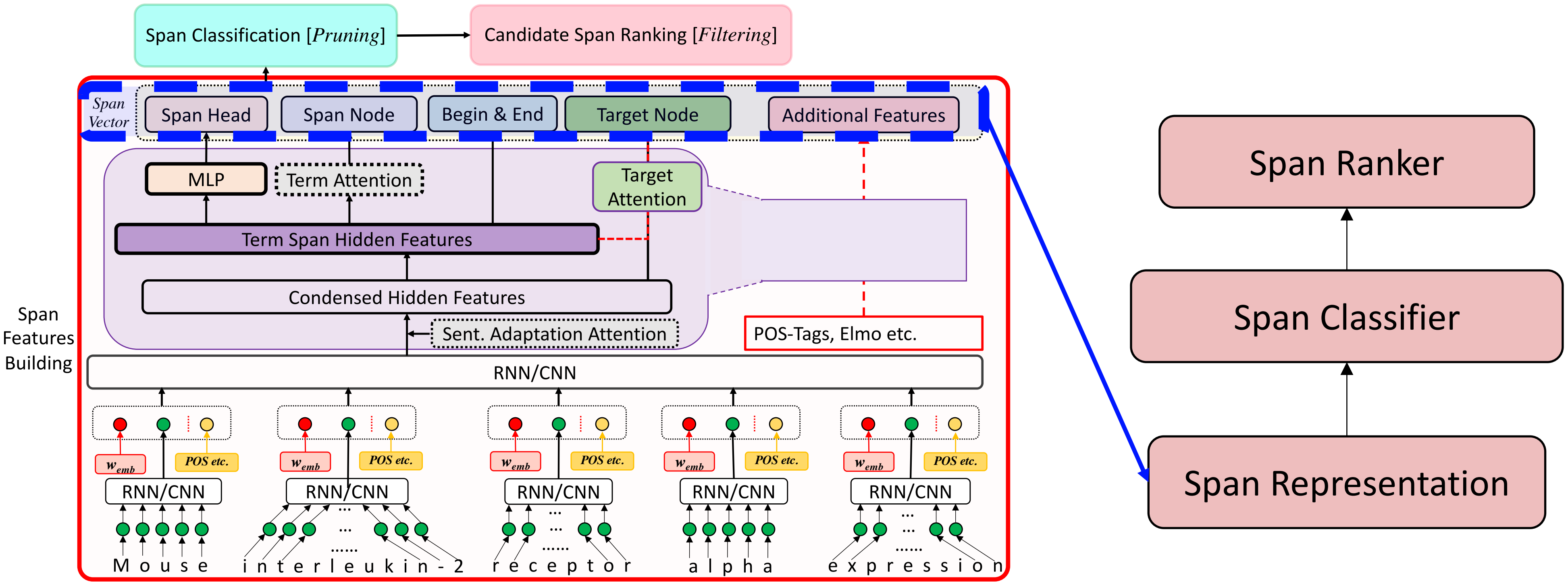
Our Model

- Architecture



Our Model

- Architecture



Filtering

Pruning

Our Model::Span Representation

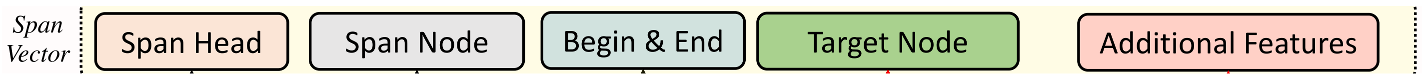
- Span (Mention) Representation S_M

Our Model::Span Representation

- Span (Mention) Representation S_M
  - S_M is formed from several Designed Feature Patterns

Our Model::Span Representation

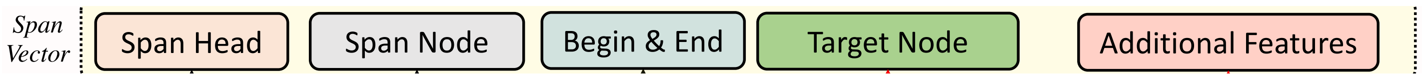
- Designed Feature Patterns



  - Span Head: is designed to contain the head word information if any and whether all the words in span can form a complete Noun Phrase

Our Model::Span Representation

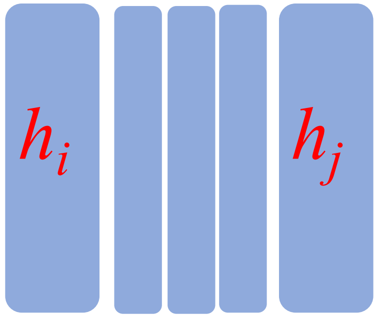
- Designed Feature Patterns



  - Span Head: is designed to contain the head word information if any and whether all the words in span can form a complete Noun Phrase

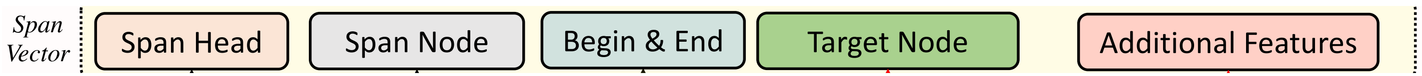
Term attention score in span: $$P_h^{[x]} = \frac{h_x * \mathbf{v_t^T}}{\sum_{x=i}^{j} h_k * \mathbf{v_t^T}} \qquad (h_x, h_k \in \boxed{H_m})$$

Vectors of Span tokens



Our Model::Span Representation

- Designed Feature Patterns

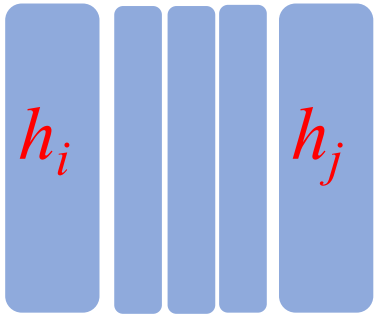


  - Span Head: is designed to contain the head word information if any and whether all the words in span can form a complete Noun Phrase

Term attention score in span:

$$P_h^{[x]} = \frac{h_x * \mathbf{v_t^T}}{\sum_{x=i}^{j} h_k * \mathbf{v_t^T}} \qquad (h_x, h_k \in \boxed{H_m})$$

Vectors of Span tokens

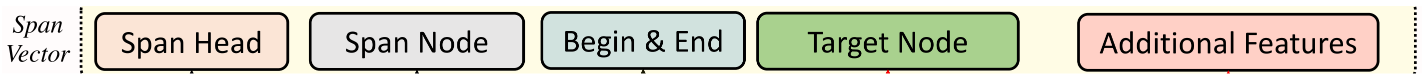


Token Vector Sum_Up:

$$V_h = \sum_{x=i}^{j} h_x * P_h^{[x]} \qquad (h_x \in H_m)$$

Our Model::Span Representation

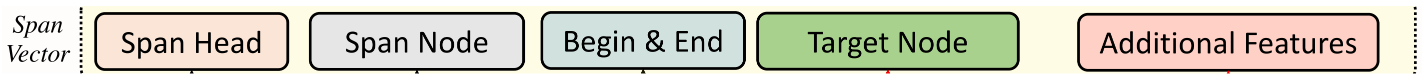
- Designed Feature Patterns



  - Span Head
  - Span Node: is designed to concentrate and contain the continuous information of the token span sequence.

Our Model::Span Representation

- Designed Feature Patterns



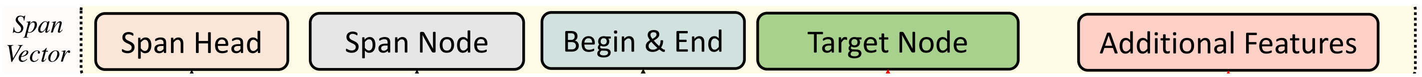
  - Span Head
  - Span Node: is designed to concentrate and contain the continuous information of the span token sequence.

Reduce the concatenation of token vectors via a Multi-Layer Perceptron:

$$V_n = MLP([h_i : h_{i+1} : ... : h_j])$$

Our Model::Span Representation

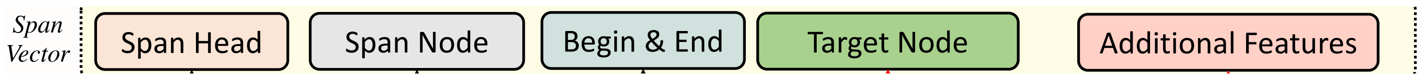
- Designed Feature Patterns



  - Span Head | Span Node
  - Begin&End: is designed to contain the feature information of begin and end word of the token span. For example, the term cannot start and end with a PREP word.

Our Model::Span Representation

- Designed Feature Patterns

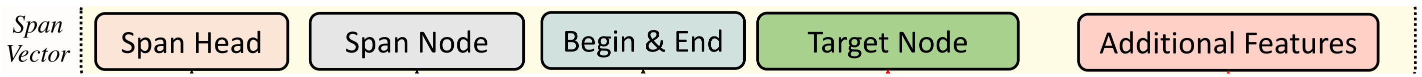


  - Span Head | Span Node
  - Begin&End: is designed to contain the feature information of begin and end word of the token span. For example, the term cannot start and end with a PREP word.

$$V_{be} = [h_i : h_j]$$

Our Model::Span Representation

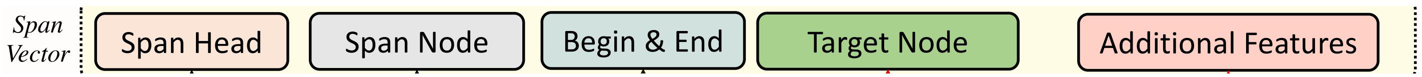
- Designed Feature Patterns



  - Span Head | Span Node | Begin&End
  - Sentence Targeted Attention Node: is designed to embed some feature information like whether the candidate span can express a concept to the complete sentence and leverage the information from the sentence level into term spans.

Our Model::Span Representation

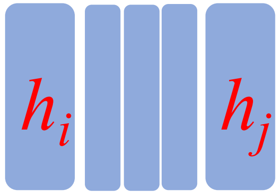
- Designed Feature Patterns



  - Span Head | Span Node | Begin&End
  - Sentence Targeted Attention Node: is designed to embed some feature information like whether the candidate span can express a concept to the complete sentence and leverage the information from the sentence level into term spans.

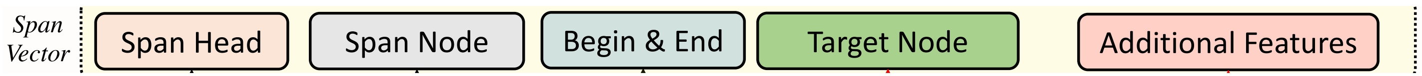
Mean the span token vectors:

$$\hat{h}_m = \sum_{x=i}^{j} h_x \quad (h_x \in H_m)$$



Our Model::Span Representation

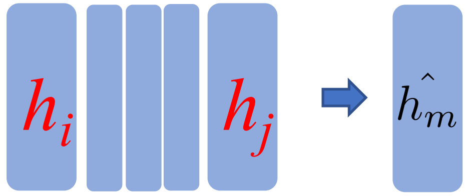
- Designed Feature Patterns



  - Span Head | Span Node | Begin&End
  - Sentence Targeted Attention Node: is designed to embed some feature information like whether the candidate span can express a concept to the complete sentence and leverage the information from the sentence level into term spans.

Mean the span token vectors:

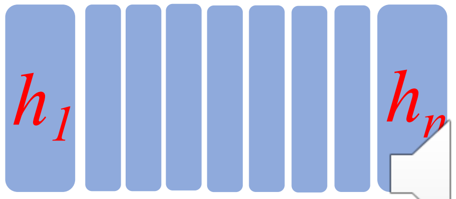
$$\hat{h}_m = \sum_{x=i}^{j} h_x \quad (h_x \in H_m)$$



Use the mean vector as target, apply attention mechanism over sentence

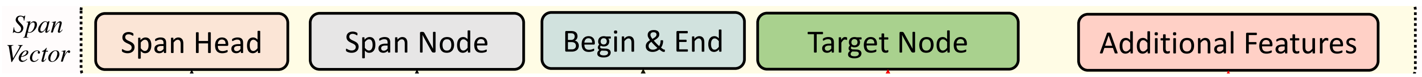
$$P_s^{[x]} = \frac{h_s[x] * \hat{h}_m^{\,T}}{\sum_{k=1}^{n} h_s[k] * \hat{h}_m^{\,T}} \quad (h_s[x], h_s[k] \in H_s)$$

$$V_s = \sum_{i=1}^{n} h_s[x] * P_s^{[x]}$$



Our Model::Span Representation

- Designed Feature Patterns



  - Span Head | Span Node | Begin&End | Sentence Targeted Attention Node
  - Length Embedding: is to convey the span length information

Our Model::Span Representation

- Designed Feature Patterns
  - Span Head | Span Node | Begin&End | Sentence Targeted Attention Node | Length Embedding

Our Model::Span Representation

- Designed Feature Patterns
  - Span Head | Span Node | Begin&End | Sentence Targeted Attention Node | Length Embedding

- Concatenation of Feature Patterns

$$S_M = [V_n, V_h, V_{be}, V_s, V_l]$$

Our Model::Span Representation

- Designed Feature Patterns
  - Span Head | Span Node | Begin&End | Sentence Targeted Attention Node | Length Embedding

- Concatenation of Feature Patterns

$$S_M = [V_n, V_h, V_{be}, V_s, V_l]$$

Span Head | Span Node | Begin&End | Sentence Targeted Attention Node | Length Embedding

Our Model::Classifier

- Span Classifier:

Get Candidates: $$\boxed{TF_G} = CLF_{FC}(\boxed{S_M})$$

Our Model::Classifier

- Span Classifier:

Get the spans classified as True Term

Get Candidates: $$\boxed{TF_G} = CLF_{FC}(\boxed{S_M})$$

Span Representation

Our Model::Classifier | Ranker

- Span Classifier:

Get the spans classified as True Term

Get Candidates: $$\boxed{TF_G} = CLF_{FC}(\boxed{S_M})$$

Span Representation

- Span Ranker:

Get Candidate Scores: $$R_{scores} = \{REG(\boxed{S_M^{T_i}}), \quad S_M^{T_i} \in S_M^{\boxed{T_G}}\}$$

Our Model::Classifier | Ranker

- Span Classifier:

Get the spans classified as True Term

Get Candidates:

$$\boxed{TF_G} = CLF_{FC}(\boxed{S_M})$$

Span Representation

- Span Ranker:

True Terms' Span Representation

Get Candidate Scores:

$$R_{scores} = \{REG(\boxed{S_M^{T_i}}), \quad S_M^{T_i} \in S_M^{\boxed{T_G}}\}$$

The spans classified as True Term

Our Model::Classifier | Ranker

- **Span Classifier:**

  Get Candidates: $$TF_G = CLF_{FC}(S_M)$$

  Get the spans classified as True Term

  Span Representation

- **Span Ranker:**

  Get Candidate Scores: $$R_{scores} = \{REG(S_M^{T_i}), \quad S_M^{T_i} \in S_M^{T_G}\}$$

  True Terms' Span Representation

  The spans classified as True Term

  Ranking the Scores: $$TM_S = RANKER|_{n=1}^{K}(R_{scores})$$

Our Model::Classifier | Ranker

- **Span Classifier:**

Get the spans classified as True Term

Get Candidates: $$\boxed{TF_G} = CLF_{FC}(\boxed{S_M})$$

Span Representation

- **Span Ranker:**

True Terms' Span Representation

Get Candidate Scores: $$\boxed{R_{scores}} = \{REG(\boxed{S_M^{T_i}}), \quad S_M^{T_i} \in S_M^{\boxed{T_G}}\}$$

The spans classified as True Term

Ranking the Scores: $$TM_S = RANKER|_{n=1}^{K}(\boxed{R_{scores}})$$

$$\mathrm{K} = \alpha \cdot |TotalWords|$$

Our Model::Training Loss

- $Loss_{(classifier)} = -(y * log(p) + (1 - y) * log(1 - p))$

Our Model:: Training Loss

- $Loss_{(classifier)} = -(y * log(p) + (1 - y) * log(1 - p))$

- $Loss_{(ranker)} = \sum_{y \in Y_{\{gold\}}} (1 - Sigmoid(y)) + \sum_{y' \in Y_{\{K-gold\}}} Sigmoid(y')$

Outline

- Background and Methods
- Our Model
- Experiments and Results
- Conclusion

Experiments and Results

- Data:
  - GENIA 3.02[1] Corpus

1: http://www.geniaproject.org/genia-corpus

Experiments and Results

- Results:

		Precision	Recall	F1
Wang et al. [10]		0.647	0.780	0.707
Yuan et al. [5]		**0.7466**	0.6847	0.7143
Our Model (Classifier)	Random Embedding	0.5044	**0.9639**	0.6622
	GloVe	0.5093	**0.9557**	0.6575
	+ POS-tag	0.5198	**0.9632**	0.6753
	+ ELMo	0.5220	**0.9541**	0.6748
	+ ALL	0.5163	**0.9698**	0.6738
Our Model (Ranker)	Random Embedding	0.7237	0.8343	0.7751
	GloVe	0.7244	0.8356	0.7760
	+ POS-tag	**0.7265**	**0.8375**	**0.7780**
	+ ELMo	0.7252	**0.8386**	0.7778
	+ ALL	**0.7316**	0.8327	**0.7789**

Experiments and Results

- Results on different feature patterns:

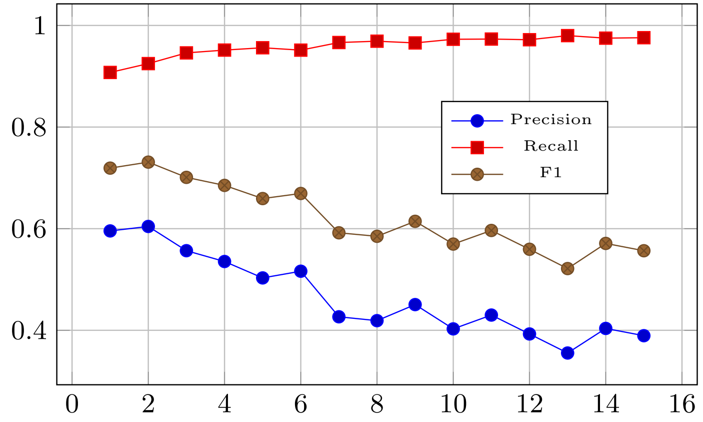
	Classifier **(F1)**	Ranker **(F1)**
Begin&End	0.6171	0.7423
+SpanLen	0.6244	0.7489
+SpanNode	0.6343	0.7535
+SpanHead	0.6488	0.7648
+TargetNode	0.6622	0.7751

Local Level VS. Sentence level

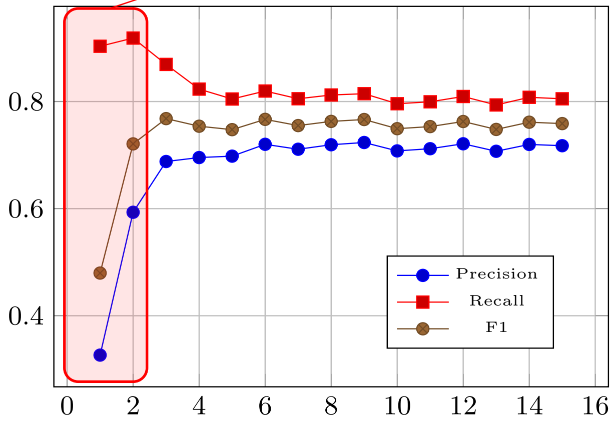
Experiments and Results

- Results on span length:

Low Precision high recall due to $K = \alpha \cdot |TotalWords|$



Classifier on lengths(Testset)



Ranker on lengths(Testset)

Example

Sentence	*Analysis of the biochemical$_3$ and cell$_5$ biological$_6$ properties$_7$ of these HSFs$_{10}$ reveals that HSF3 has properties in common with both HSF1 and HSF2 and yet has features which are distinct from both .*	*Using a$_1$ polyclonal$_2$ antibody$_3$ to murine$_5$ NFATp$_6$, Western blot analysis of various$_{12}$ mouse$_{13}$ tissues$_{14}$ demonstrated that the 110-130-kDa$_{18}$ NFATp$_{19}$ protein$_{20}$ was highly expressed in thymus and spleen .*
Gold	[3, 3], [3, 7], [5, 6], [7, 7], [10, 10], [13, 13], [15, 15], [20, 20], [22, 22]	[5, 6], [6, 6], [8, 10], [13, 13], [13, 14], [19, 19], [19, 20], [25, 25], [27, 27]
Classifier	[3, 3], [3, 7], [3, 5], [5, 5], [5, 6], [5, 7], [6, 7], [10, 10], [13, 13], [13, 15], [20, 20], [22, 22]	[2, 3], [5, 5], [5, 6], [6, 6], [8, 10], [12, 14], [13, 13], [13, 14], [18, 18], [18, 20], [18, 19], [19, 19], [19, 20], [25, 25], [27, 27]
Ranker	[20, 20], [13, 13], [22, 22], [10, 10], [3, 7], [6, 7	[19, 20], [12, 14], [18, 20], [19, 19], [13, 14], [6, 6], [2, 3], [25, 25], [27, 27], [5, 6], [18, 19]

Outline

- Background and Methods
- Our Model
- Experiments and Results
- Conclusion

Conclusion

- This method can achieve the state-of-art results without any external features. (Refer slides 48)
- In contrast with local features, sentence level features can contribute more in term extraction task. (Refer slides 49)
- Designing reasonable feature pattern to reform hidden features is more efficient.

Thank you

Code is available at: https://github.com/CooDL/Nested-Term-Extraction