A Minwise Hashing Method for Addressing Relationship Extraction from Text

David S. Batista, Rui Silva, Bruno Martins and Mário J. Silva

Instituto Superior Técnico and INESC-ID, University of Lisbon, Portugal

Relationship Extraction from Text

- Detection and classification of semantic relations between pairs of entities:

  Nanjing is the capital of **Jiangsu** province. ***(LOCATION*** *capital-of* ***LOCATION)***

  Jimi Hendrix was born in **Seattle** in 1942. ***(PERSON*** *place-of-birth* ***LOCATION)***

- **Domain-specific**: known relations *a priori*
  - Supervised
  - Features : lexical, syntactic and semantic information
  - Kernel : explore input representations exhaustively
    - No need to explicitly representing the features
    - CPU and memory demanding!

- **Open-domain:** unknown relations *a priori*
  - Unsupervised (based on hand-made rules)
  - Large datasets

The Relationship Extraction Task

"**Joe** said that **Margaret Thatcher** died on the morning of 8 April in **London** after suffering a stroke."

- Possible relations:
  - Joe → Margaret Thatcher → NOT-RELATED
  - Margaret Thatcher → Joe → NOT-RELATED
  - Joe → London → NOT-RELATED
  - London → Joe → NOT-RELATED
  - Margaret Thatcher → London → DEATH-PLACE
  - London → Margaret Thatcher → PLACE-OF-DEATH
- Classify pairs of named entities according to the type of semantic relation class

Proposed Method

- New method based on **weighted *kNN* classification**
  - Supervised
  - Scalable (not CPU or memory demanding) but still achieves competitive accuracy
  - Based Jaccard similarity between relation instances
    - Use min-hash to approximate Jaccard similarity
    - Use locality sensitive hashing to find *kNN* instances
  - Classification based on weighted votes from *kNN* instances

Outline

- ~~Introduction: Relationship Extraction Task~~
- ~~Proposed Method~~

1. Representing Relations Instances as Features
2. Min-Hash/Locality-Sensitive Hashing
3. Complete process: Indexing / Classification
4. Experiments: Datasets and Results
5. Scalability: Indexing and Classification
6. Conclusions and Future work

Representing Relation Instances

“**Joe** said that **Margaret Thatcher** died on the morning of 8 April in **London** after suffering a stroke.”

- The relation instance **Margaret Thatcher** → **London** can be represented as follows:

  - **BEFORE-BETWEEN** – tokens before and between the related entities
    - **Joe** said that **Margaret Thatcher** died on the morning of 8 April in

  - **BETWEEN** – tokens between the related entities
    - died on the morning of 8 April in

  - **BETWEEN-AFTER** – tokens between and after the related entities
    - died on the morning of 8 April in **London** after suffering a stroke.

The Considered Features

- Characters n-grams of size 4
- Lexical features extracted with the *MorphAdorner* package:
  - Verbs (normalized)
  - Prepositions (e.g., *between*, *above*, *within, etc.*)
  - Verbs in the past participle (passive voice can indicate direction of relation)
    - *Harry ate six shrimps at dinner. (active)*
    - *At dinner, six shrimps were eaten by Harry. (passive)*
  - ReVerb pattern - Verbs, followed by nouns, adjectives or adverbs, and ending in a preposition (e.g., *was the guitar player of, died on the*)

- Window of 3 tokens for BEFORE-BETWEEN and BETWEN-AFTER
- Features (e.g., tokens) from each of the three groups are considered different
- Also experimented : VerbNet, WordNet, Levin verb classes
  - No major improvements on the results and more time to process ...

Outline

- ~~Introduction: Relationship Extraction Task~~
- ~~Proposed Method~~

1. ~~Representing Relations Instances as Features~~
2. Min-Hash/Locality-Sensitive Hashing
3. Complete process: Indexing / Classification
4. Experiments: Datasets and Results
5. Scalability: Indexing and Classification
6. Conclusions and Future work

Min-Hash

- Technique for quickly estimating how similar two sets are.
- Invented by Andrei Broder (1997) and initially used in the AltaVista search engine to detect duplicate web pages and eliminate them from search results.
- Gives an approximation of Jaccard similarity measure between two given sets.

$$\mathrm{J}(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{|S_1 \cap S_2|}{|S_1| + |S_2| - |S1 \cap S_2|}$$

Min-Hash Signatures

- *hash_func(x)* maps members of *S1* and *S2* to distinct integers
- *h_min(S)* is the member *x* of *S* with the minimum value of h(x)
- The probability that both sets share the same minimum hash value is equal to the ratio of their common elements to their total elements

$$\pi : \Omega \longrightarrow \Omega \quad , \text{ where } \quad \Omega = \{1, 2, \ldots, D\}.$$

$$\Pr\left(\min(\pi(S_1)) = \min(\pi(S_2))\right) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \mathrm{J}(S_1, S_2)$$

- If *r* is a random variable that is 1 when h_min(A) = h_min(B) and zero otherwise, then *r* is an unbiased estimator of J(S1,S2)

$$\hat{\mathrm{J}}(S_1, S_2) = \frac{1}{k} \sum_{j=1}^{k} 1(\min(\pi_k(S_1)) = \min(\pi_k(S_2)))$$

Locality-Sensitive Hashing for Min-Hash Signatures

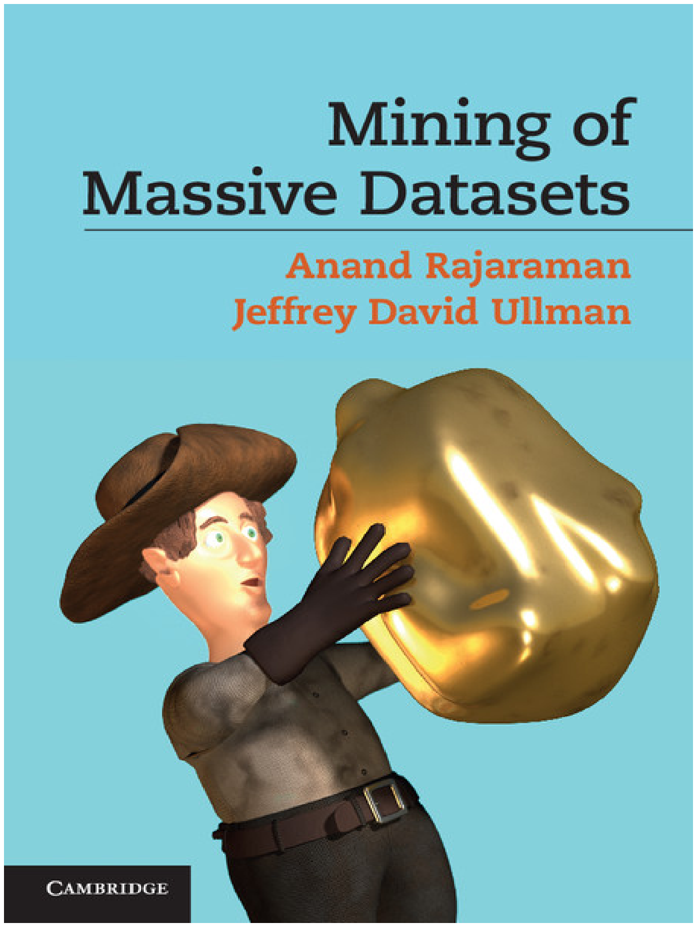
- Instances represented as min-hash signatures:

  [h1_min(*S1*), h2_min(*S1*), ..., hk_min(*S1*)]

- Divide each signature into *L* bands
  - Band 1 = hash ([h1_min(*S1*), h2_min(*S1*)])
  - Band 2 = …
  - Band L = hash ([hk-1_min(*S1*), hk_min(*S1*)])

- Each band corresponds to an *n*-tuple from the min-hash signatures
- An index is built with *L* different hash tables, each corresponding to an *n*-tuple from the min-hash signatures
- Candidate similar instances are those with a band in common

Confusing ?



- “Mining of Massive Datasets” (Rajaraman, Anand, and Jeffrey David Ullman. Cambridge University Press, 2012)

  → “Chapter 3: Finding Similar Items”

- Available for free at:

  http://infolab.stanford.edu/~ullman/mmds.html

  (much more: graph analysis, clustering, dimensionality reduction, etc.)

Outline

- ~~Introduction: Relationship Extraction Task~~
- ~~Proposed Method~~

1. ~~Representing Relations Instances as Features~~
2. ~~Min-Hash/Locality-Sensitive Hashing~~
3. Complete process: Indexing / Classification
4. Experiments: Datasets and Results
5. Scalability: Indexing and Classification
6. Conclusions and Future work

Indexing

- Given a set of annotated relation instances
  - Generate groups:
    - BEFORE-BETWEEN
    - BETWEEN
    - BETWEEN-AFTER
  - Extract features
  - Calculate min-hash signatures
  - Min-hash signatures splitted and hashed into *L* bands

Classification

- Given a new sentence with tagged named entities:
  - Generate groups:
    - BEFORE-BETWEEN
    - BETWEEN
    - BETWEEN-AFTER
  - Extract features
  - Calculate min-hash signatures
  - Indexed relationships instances with at least one common band are candidates
  - Estimate Jaccard similarity with the available min-hash signatures
- Each of the *kNN* nearest neighbours votes with his semantic class
- Each vote is weighted according to the similarity towards the instance
- The semantic class with the higher score is assigned

Weighted *kNN* classification

- Given an new instance X to be classified and the top-5 more similar relations:

  - 1st *place-of-birth* (0.53)
  - 2nd *place-of-death* (0.48)
  - 3rd *place-of-death* (0.45)
  - 4th *place-of-death* (0.42)
  - 5th *place-of-death* (0.41)

- X will be classifed as *place-of-death,* which is more frequent, higher vote.

Outline

- ~~Introduction: Relationship Extraction Task~~
- ~~Proposed Method~~

1. ~~Representing Relations Instances as Features~~
2. ~~Min-Hash/Locality-Sensitive Hashing~~
3. ~~Complete process: Indexing / Classification~~
4. Experiments: Datasets and Results
5. Scalability: Indexing and Classification
6. Conclusions and Future work

Experiments

- Test with 3 different datasets/domains
  - SemEval 2010 – relations between nominals
  - AImed – interactions between human proteins
  - Wikipedia – relations between named entities

- Varying 3 parameters:
  - Size of min-hash signatures
  - Number of bands in LSH
  - Number of *k* nearest neighbours

Datasets - Statistical Characterization

	SemEval		Wikipedia		AImed
	Train	Test	Train	Test	Data
# Sentences	8,000	2,717	2,199	926	2,202
# Terms	137,593	46,873	49,721	20,656	75,878
# Relation classes	19	19	47	47	2
# Relation instances (except *not-related/other*)	6,590	2,263	15,963	6,386	1,000
# Nominals	16,001	5,434	5,468	2,258	4,084
Avg. sentence length (terms)	119.8	119.4	177.2	172.8	184.2
StDev. sentence length (terms)	45.0	44.4	104.5	100.1	98
Avg. instances/class	421	143	295.6	135.9	1,961.5
StDev. instances/class	317.5	105.5	1707.3	728.2	1,372.5
Max. instances/class (except *not-related/other*)	844	22	268	113	1,000
Min. instances/class	1	1	1	1	1,000

- **SemEval** – 10,717 sentences and 19 classes
- **Wikipedia** – 3,125 sentences and 47 classes
  - Significantly skewed in the class distribution
- **AImed** – 2,202 sentences and 2 classes

Results

Dataset	Min Hash	1 kNN			3 kNN			5 kNN			7 kNN		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
SemEval (18 classes)	200/25	0.662	0.622	0.641	0.683	0.642	0.662	0.698	0.652	0.674	0.698	0.637	0.666
	200/50	0.662	0.621	0.640	0.683	0.643	0.662	0.698	0.651	0.673	0.698	0.636	0.666
	400/25	0.664	0.636	0.650	0.685	0.668	0.676	0.708	0.672	0.690	0.691	0.667	0.679
	400/50	0.663	0.635	0.649	0.684	0.664	0.674	**0.708**	0.674	**0.690**	0.694	0.670	0.682
	600/25	0.657	0.631	0.644	0.677	0.660	0.669	0.697	0.674	0.685	0.695	0.660	0.677
	600/50	0.657	0.631	0.644	0.676	0.658	0.667	0.699	**0.678**	0.688	0.694	0.664	0.678
	800/25	0.654	0.630	0.642	0.675	0.656	0.665	0.694	0.662	0.678	0.696	0.658	0.677
	800/50	0.654	0.632	0.643	0.677	0.658	0.667	0.698	0.665	0.681	0.696	0.658	0.676
Wikipedia (46 classes)	200/25	0.410	0.336	0.369	0.434	0.335	0.378	0.439	0.310	0.363	0.489	0.323	0.389
	200/50	0.409	0.336	0.369	0.435	0.336	0.379	0.440	0.310	0.364	0.489	0.321	0.387
	400/25	0.453	0.350	0.394	0.472	0.354	0.405	0.507	0.348	0.413	0.485	0.323	0.388
	400/50	0.450	0.349	0.393	0.468	0.354	0.403	0.503	0.350	0.412	0.509	0.328	0.399
	600/25	0.419	0.344	0.378	0.439	0.352	0.391	0.492	0.364	0.419	0.522	**0.365**	**0.430**
	600/50	0.419	0.343	0.377	0.444	0.354	0.394	0.485	0.353	0.408	**0.532**	0.353	0.425
	800/20	0.416	0.344	0.377	0.431	0.348	0.385	0.493	0.351	0.410	0.513	0.343	0.411
	800/50	0.419	0.345	0.378	0.433	0.350	0.387	0.515	0.346	0.414	0.517	0.338	0.409
AImed (1 class)	200/25	0.405	0.545	0.465	0.430	0.509	0.466	0.480	0.484	0.482	0.507	0.460	0.482
	200/50	0.405	0.545	0.465	0.430	0.509	0.466	0.480	0.484	0.482	0.507	0.460	0.482
	400/25	0.420	0.589	0.491	0.451	0.554	0.497	0.481	0.524	0.501	0.516	0.502	0.509
	400/50	0.420	0.588	0.490	0.455	0.561	0.502	0.484	0.529	0.505	**0.519**	0.505	0.512
	600/25	0.409	0.605	0.488	0.445	0.571	0.500	0.475	0.529	0.500	0.511	0.513	0.512
	600/50	0.409	0.605	0.488	0.445	0.571	0.500	0.475	0.530	0.501	0.511	0.513	0.512
	800/25	0.416	0.613	0.496	0.453	0.595	0.514	0.481	0.547	0.512	0.490	0.512	0.501
	800/50	0.418	**0.614**	0.498	0.454	0.596	**0.515**	0.482	0.545	0.511	0.489	0.514	0.501

Results Per Class (SemEval)

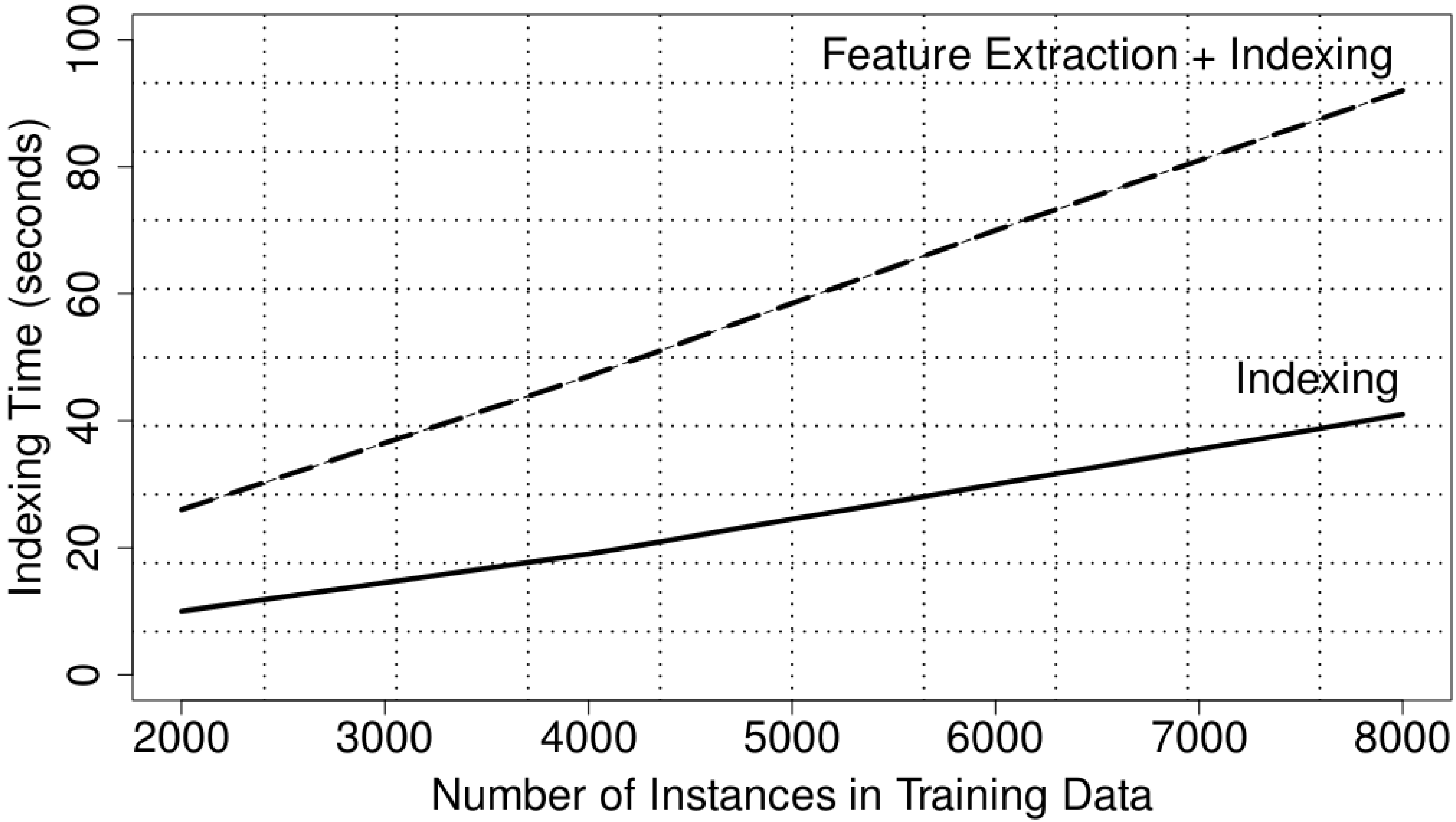
Relation	Direction	Instances (train/test)	Asymmetrical Precision	Asymmetrical Recall	Asymmetrical F1	Symmetrical Precision	Symmetrical Recall	Symmetrical F1
Cause-Effect	(e1,e2)	344/134	0.843	0.843	0.843	0.798	0.902	0.847
	(e2,e1)	659/194	0.735	0.902	0.810			
Component-Whole	(e1,e2)	470/162	0.572	0.759	0.653	0.628	0.670	0.648
	(e2,e1)	150/129	0.609	0.520	0.561			
Entity-Destination	(e1,e2)	844/291	0.744	0.911	0.819	0.747	0.901	0.817
	(e2,e1)	1/1	1.000	0.000	0.000			
Entity-Origin	(e1,e2)	568/211	0.789	0.815	0.802	0.756	0.795	0.775
	(e2,e1)	148/47	0.667	0.723	0.694			
Product-Producer	(e1,e2)	323/108	0.670	0.602	0.634	0.673	0.589	0.628
	(e2,e1)	394/123	0.654	0.569	0.609			
Member-Collection	(e1,e2)	78/32	0.778	0.438	0.560	0.767	0.777	0.772
	(e2,e1)	612/201	0.776	0.791	0.783			
Message-Topic	(e1,e2)	490/210	0.751	0.733	0.742	0.778	0.778	0.778
	(e2,e1)	144/51	0.750	0.706	0.727			
Content-Container	(e1,e2)	374/153	0.726	0.778	0.751	0.706	0.802	0.751
	(e2,e1)	166/39	0.627	0.821	0.711			
Instrument-Agency	(e1,e2)	97/22	0.429	0.545	0.480	0.605	0.667	0.634
	(e2,e1)	407/134	0.615	0.679	0.645			
Other	—	1410/454	—	—	—	0.442	0.293	0.352
Macro-average	—	—	0.708	0.674	0.690	0.718	0.764	0.740

Outline

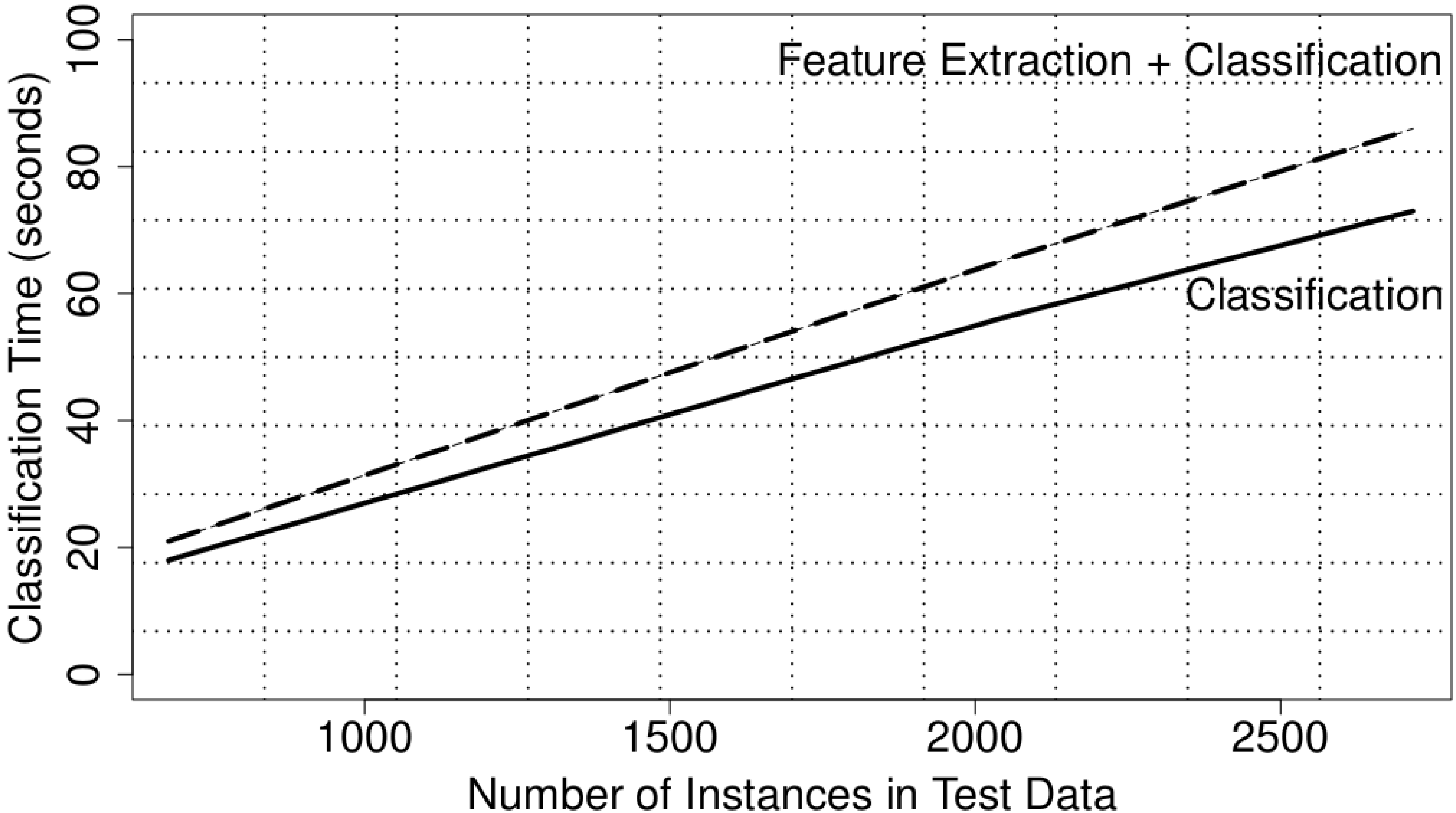
- ~~Introduction: Relationship Extraction Task~~
- ~~Proposed Method~~

1. ~~Representing Relations Instances as Features~~
2. ~~Min-Hash/Locality-Sensitive Hashing~~
3. ~~Complete process: Indexing / Classification~~
4. ~~Experiments: Datasets and Results~~
5. Scalability: Indexing and Classification
6. Conclusions and Future work

Scalability - Indexing (SemEval)



Scalability - Classification (SemEval)



Outline

- ~~Introduction: Relationship Extraction Task~~
- ~~Proposed Method~~

1. ~~Representing Relations Instances as Features~~
2. ~~Min-Hash/Locality-Sensitive Hashing~~
3. ~~Complete process: Indexing / Classification~~
4. ~~Experiments: Datasets and Results~~
5. ~~Scalability: Indexing and Classification~~
6. Conclusions and Future work

Conclusions

SemEval 2010 F1 scores:

- Best system: 0.82
- Second best: 0.77
- Median score: 0.68
- **Our best score: 0.69 (172s feature extraction + indexing + classification)**
- Participating systems used extra resources:
  - Google's n-gram, Cyc, WordNet, Roget's Taxonomy, or Levin's verb classes

- **AImed dataset F1 scores:**
  - Sub-sequence kernel from Bunescu and Mooney: 0.54
  - All-dependency-paths kernel from Airola et al.: 0.56 (4 521 seconds)
  - **Our best score 0.52 (161s feature extraction + indexing + classification)**

- **Linear Scalable**
- **Not CPU or memory demanding**
- **Still achieves competitive accuracy**

Conclusions

- Advantages over kernel methods:
  - **Simple**:
    - mostly based on extracting n-grams and POS tags
    - (almost) language independent, needs POS-tagger
  - **Online:** to consider new training examples, we only need to compute their min-hash signatures and index
  - **Scalable:** *kNN* search is made efficiently

Future Work

- Experiments with more datasets
  - Dataset from the ACE evaluation campaign
- Other similarity search techniques
  - Graph-based representations (from lexical information and from constituency/dependency parsing)
  - Minwise hashing methods for comparing graphs
  - *b*-bit minwise hashing approach for improving storage efficiency on very large datasets
  - extension proposed by Chum et al. for approximating weighted set similarity measures

Thank you!

谢谢 [xie xie]

:-)

Questions?