Back to Patterns:
Efficient Japanese Morphological Analysis with Feature-Sequence Trie

Naoki Yoshinaga (Institute of Industrial Science, The University Tokyo)

Code in C++ (<1000 lines): https://www.tkl.iis.u-tokyo.ac.jp/~ynaga/jagger/ License: GPLv2, LGPLv2.1, BSD

Background: The amount of data is increasing, whereas NLP models become inefficient (larger and slower)

- The amount of **text has been increasing**
  - Microblog posts via smarphones (Twitter)
  - Online communication (zoom, slack)
- **Models** focus on accuracy and **become slower**
  - The efficient neural methods are only *relatively efficient* and much slower than classical methods

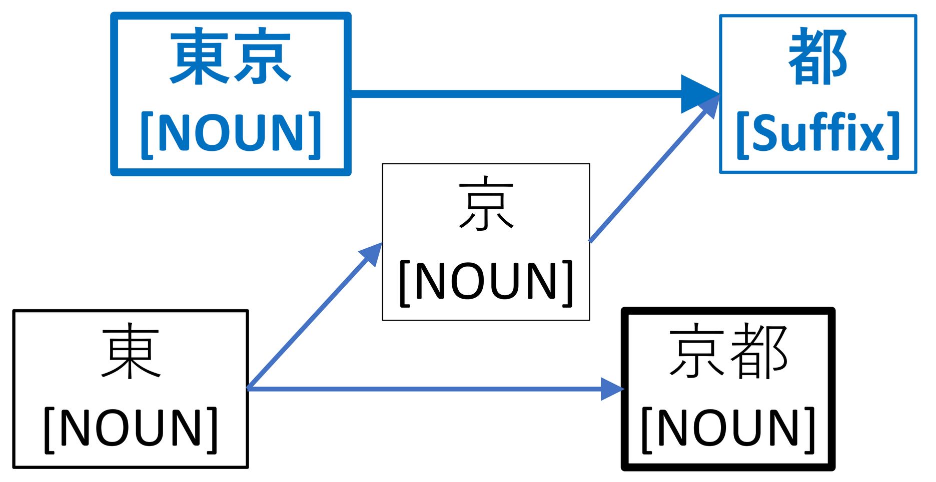
Classical methods are still used to process SNS posts in sociolinguitics and marketing

Need for speed-intensive appraoach to NLP

Target task in this paper: Japanese morphological analysis

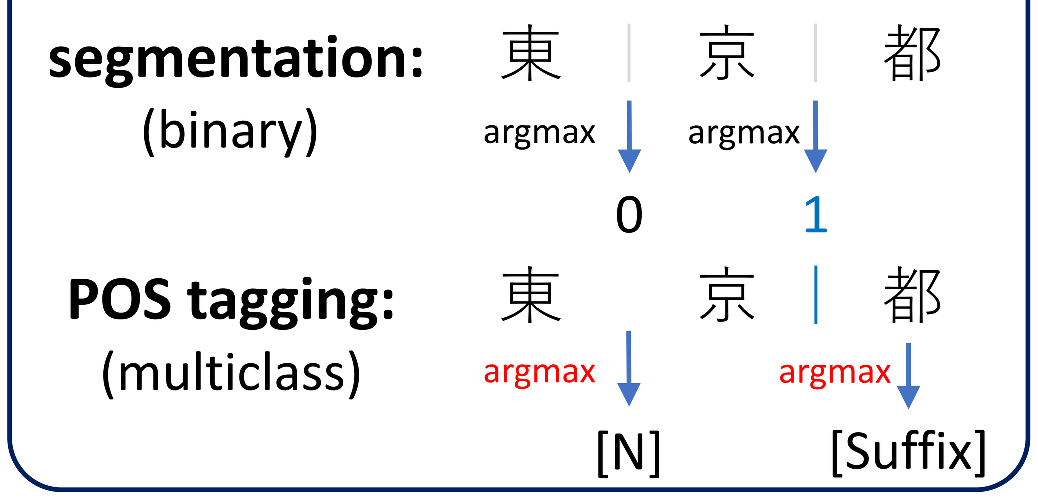
Search-based [Kudo+ 2004]

DP to find global argmax



Classification-based [Neubig+ 2011]

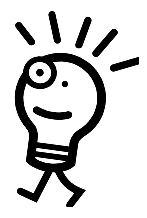
Two-stage classification



perform expensive argmax operations over class-wise feature weights

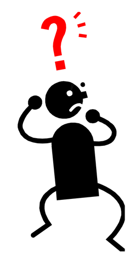
Jagger (proposal): A **remarkably simple yet accurate** pattern-based method for morphological analysis

- Let's make **the fastest method more accurate** (instead of making the most accurate method slightly more efficient) for **Japanese morphological analysis** (word segmentation, POS tagging, and lemmatization; latter two as tagging)

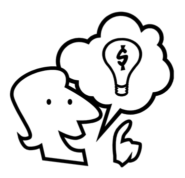


Idea:

Bypass expensive argmax operations via patterns
- Assume morphological analysis **as single multiclass classification** (where to segment and what to tag)
- **Segment and tag words by greedily applying patterns** inspired by longest-matching for word segmentation



Issue: How to obtain reliable patterns?

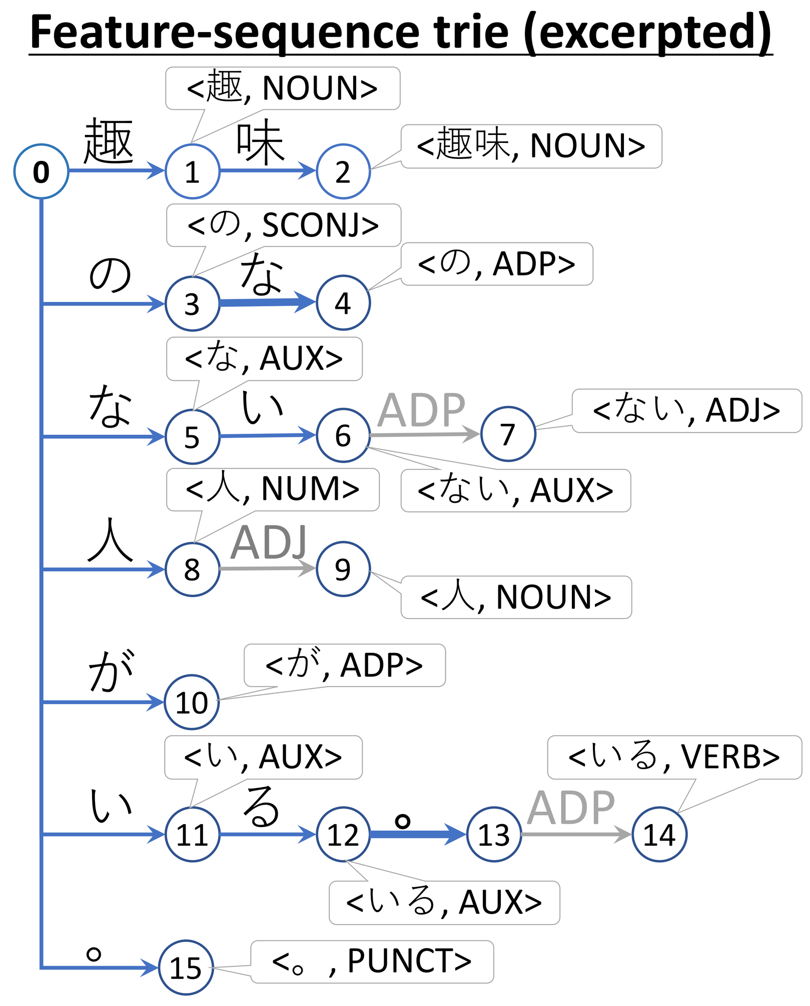


solution:

Extract pattens as learning-based methods do
- **Design a pattern template from feature templates** of learning-based methods [Kudo+ 2004, Neubig+ 2011] as:
  posterior contexts + a previous tag
- **Use the training data and a dictionary** to extract patterns by frequency (**offline argmax**)

Pattern	Word	POS (level 1)
趣味|	趣味	NOUN
の|な	の	ADP
ない|_ADP	ない	ADJ
人|_ADJ	人	NOUN
が|	が	ADP
いる|。_ADP	いる	VERB
。|	。	PUNCT

Feature-sequence trie (excerpted)



Keep only minimum patterns for the same segmentation offsets and tags to avoid extra matching to features

Evaluation: Jagger can process 1,000,000 sents/s with accuracy comparable to learning-based baselines

We compare Jagger (proposal) with the learning-based baselines using two dictioinaries on two common datasets
- **Baselines**: search-based method [Kudo+ 2004] (**MeCab 0.996, Vibrato 0.5.0**), classification-based method [Neubig+ 2011] (**Vaporetto 0.6.2**)
- **Dictonaries**: mecab-jumandic-5.1 (475,716 words) and mecab-jumandic-7.0 (702,358 words; augmented from Wikipedia words)

Method	Kyoto (Kyoto-University Text Corpus, newspaper)								KWDLC (Kyoto-University Web Documents Lead Corpus, Web)							
	dict: jumandic-5.1				dict: jumandic-7.0				dict: jumandic-5.1				dict: jumandic-7.0			
	speed [sents/s]	mem [MiB]	seg (F_1)	POS (F_1)	speed [sents/s]	mem [MiB]	seg (F_1)	POS (F_1)	speed [sents/s]	mem [MiB]	seg (F_1)	POS (F1)	speed [sents/s]	mem [MiB]	seg (F_1)	POS (F_1)
MeCab	66,455	55.81	98.68	95.97	59,453	77.98	98.37	96.10	92,110	53.88	97.13	94.30	81,598	76.38	97.99	95.62
Vibrato	142,983	97.75	-	-	111,367	164.20	-	-	190,703	97.92	-	-	146,235	163.99	-	-
Vaporetto	117,767	658.80	98.94	96.92	105,316	828.85	99.08	97.05	200,823	642.63	97.35	94.08	174,900	842.40	97.53	94.68
Jagger	**1,007,344**	**26.39**	**98.73**	**96.55**	**974,316**	**35.09**	**98.68**	**96.57**	**1,524,305**	**28.89**	**97.17**	**94.20**	**1,503,424**	**40.22**	**97.60**	**94.63**

Jagger is 7-16x faster than baselines with 1/2 to 1/20 as much memory, while achieving comparable accuracy

Method (word segmentation)	KWDLC (Web)	
	speed [sents/s]	mem [MiB]
MeCab	62,495	40.52
Vibrato	121,375	163.92
Vaporetto	366,119	283.49
Jagger	1,942,477	21.05
Sentencepiece, 8k	150,962	9.05

Method	KWDLC→Kyoto		Kyoto→KWDLC	
	seg (F_1)	POS (F_1)	seg (F_1)	POS (F_1)
MeCab	97.90	94.82	97.78	94.48
Vaporetto	95.76	91.31	97.05	92.72
Jagger	97.25	93.30	97.22	93.12

Jagger is the fastest in segmentation and accurate in cross-domain settings

Method	Kyoto				KWDLC			
	speed [sents/s]	mem [MiB]	seg (F_1)	POS (F_1)	speed [sents/s]	mem [MiB]	seg (F_1)	POS (F_1)
Juman++v2*	5384	300.80	99.37	97.74	7753	290.05	98.37	96.42
Jagger	974,316	35.09	98.68	96.57	1,503,424	40.22	97.60	94.63

180x faster than sota neural method, with 1-2% loss in accuracy
(* Juman++ uses more dictionary words and RNN trained on Web text)

Message to researchers

Because the **accuracies are becoming saturated** on NLP benchmark datasets with a larger foundation model, researchers may want to **set diverse goals based on underrepresented metrics besides accuracy (e.g., efficiency)**