Natural Language Processing

Prof. Dr. Jannik Strötgen
jannik.stroetgen@h-ka.de

Summer 2024

Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Informatik und
Wirtschaftsinformatik

Preliminary Agenda

Date	Topic
19.03.2024	Organisation & motivation
26.03.2024	Introduction to NLP and
02.04.2024	*no lecture (Easter)*
09.04.2024	Pre-Processing and Part-of-Speech Tagging
16.04.2024	Parsing
23.04.2024	Named Entity Recognition and Disambiguation
30.04.2024	Similarity and Search
07.05.2024	Language Models: Static Word Embeddings

Date	Topic
14.05.2024	Contextual Embeddings
21.05.2024	*no lecture (Whitsun break)*
28.05.2024	Text Mining and Sentiment Analysis
04.06.2024	Information Extraction & QA
11.06.2024	Applications exploiting NLP
18.06.2024	NLP with LLMs
25.06.2024	My Research Topics
02.07.2024	Recap, exam preparation

[3] Part-of-Speech Tagging

Outline

[3] Part-of-Speech Tagging

1. Parts of speech (POS)
2. Tagging
3. Feature-based POS tagging
4. Probabilistic POS tagging
   - Unigram tagger
   - N-gram tagger
5. Rule-based POS tagging
   - Brill tagger

From the Introduction

2. Part-of-Speech (POS) Tagging

Adding information about word types helps in determining the roles that words play and in understanding the meaning of a text. For example, we can identify verbs, adverbs, nouns, determiner (articles), etc.

Ah, distinctly I remember it was in the bleak December;
And each separate dying ember wrought its ghost upon the floor.
Eagerly I wished the morrow;—vainly I had sought to borrow
From my books surcease of sorrow—sorrow for the lost Lenore—
For the rare and radiant maiden whom the angels name Lenore—
Nameless here for evermore.

March 19, 2024 Natural Language Processing @ HKA - Prof. Dr. Jannik Strötgen 11

Parts of Speech (POS)

Let’s Disambiguate Some Homonyms!

The **die** is **fair**
but the house wins.

You will **die** at the **fair**
if you enter the house.

Homonyms and Parts of Speech

The **die** is **fair** but the house wins.
noun adjective

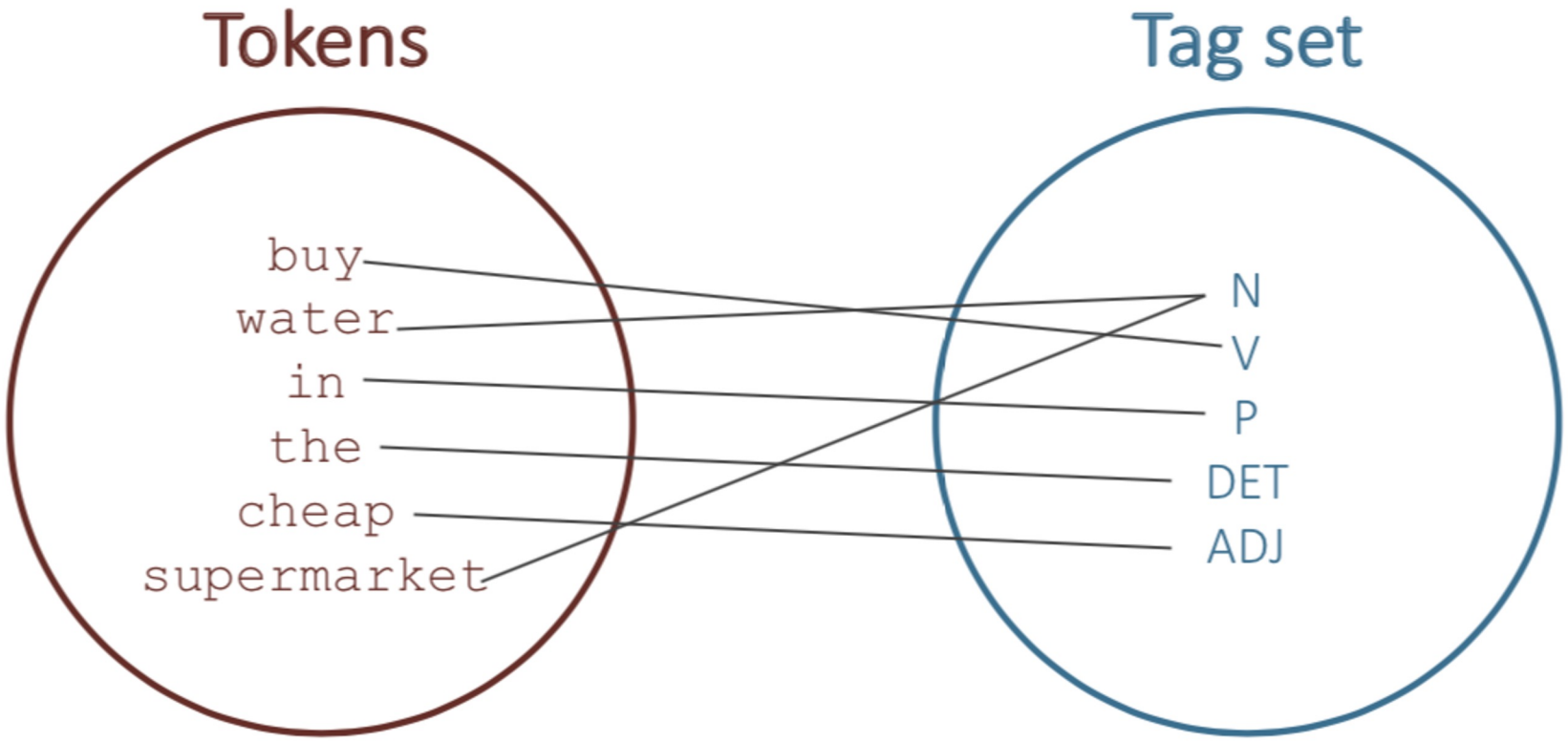
You will **die** at the **fair** if you enter…
verb noun

Homonyms and Parts of Speech

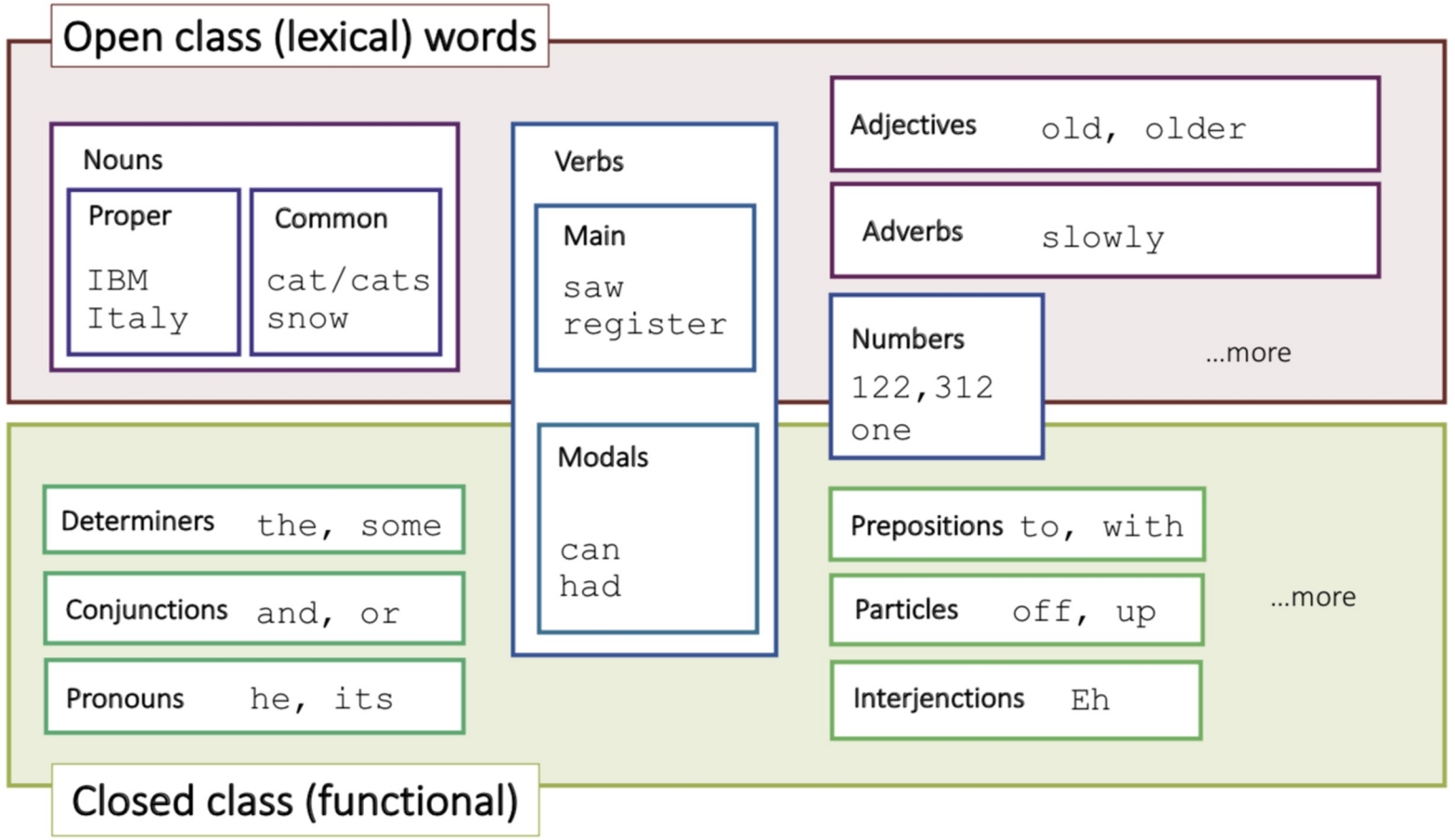
```
The die is  fair but the house wins.
DT  NN  VBZ  JJ  CC  DT   NN   VBZ
```


```
You will die at the fair if you enter…
PRP  MD  VB  IN DT   NN  IN PRP VBP
```

Part of Speech Tagging



Part of Speech Hierarchy



Part of Speech Tagging Task

Words often have more than one POS

- `The` **`die`** `is cast` → **NN**
- `To` **`die`** `or not to die` → **VB**
- The POS tagging problem addresses how to determine the POS tag for a particular instance of a word.

Example:

- Input: `The die is fair`
- Ambiguity: `DT NN/VB VBZ JJ/NN`
- Output: `The[DT] die[NN] is[VBZ] fair[NN]`

Tagging

The Purpose of POS Tagging

1. Collapsing distinctions into equivalence classes
   - All personal pronouns (`You`, `they`, etc.) tagged with `PRP`
2. Introducing distinctions to resolve ambiguities
   - E.g., `leaves` could be tagged with `NN` or `VB`
3. Helpful features in classification and prediction tasks
   - **Named entity extraction**:
     `Essen` could be the name of a German city, but not if it is tagged `VB`
   - **Machine translation**:
     Are translated sentences syntactically correct?
     (stay tuned for syntactic parsing next week!)

Terminology

Tagging

- The process of associating labels with each token in a text
- Not limited to parts of speech (e.g., helpful in sentiment analysis)

Tags

- The labels that are assigned to the tokens

Tag Set

- The collection of tags that is used for a particular task
- Tag sets are defined by linguists and not unambiguous.
- Different languages require different language-specific tag sets

→ It is difficult to compare tags between different languages

Tagset Requirements

Informativeness-related parameters:

- The size of the tagset
- The degree of the ambiguity

→ Does the tagset represent all unique cases in the language, or does ambiguity still exist?

Specifiability

- Inter-annotator (interjudge) agreement should be good

→ What is the agreement when multiple (trained) linguists tag the same text?

Tagset Requirements: Example

Collapsing parts of the Penn Treebank tagset:

- `JJ:  adjective` example: `smart`
- `JJR: adjective (comparative)` example: `smarter`
- `JJS: adjective (superlative)` example: `smartest`
- Collapse all three tags into one: `JJ`

What happens to...

- informativeness? Size of the tagset decreases. Ambiguity increases.
- specifiability? Inter-annotator agreement likely increases.

Frequently Used Tagset

English:

- Penn Treebank: 45 tags
- Brown corpus: 87 tags
- Lancaster UCREL C5: 61 tags (used to tag the BNC)
- Lancaster C7: 145 tags

German:

- Stuttgart-Tübingen Tagset (STTS): 54 tags

Penn Treebank Tagset

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential there
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun

Number	Tag	Description
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	to
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

Stuttgart-Tübingen Tagset (STTS)

ADJA	attributives Adjektiv	[*das*] grosse [*Haus*]
ADJD	adverbiales oder prädikatives Adjektiv	[*er fährt*] schnell [*er ist*] schnell
ADV	Adverb	schon, bald, doch
APPR	Präposition; Zirkumposition links	in [*der Stadt*], ohne [*mich*]
APPRART	Präposition mit Artikel	im [*Haus*], zur [*Sache*]
APPO	Postposition	[*ihm*] zufolge, [*der Sache*] wegen
APZR	Zirkumposition rechts	[*von jetzt*] an
ART	bestimmter oder unbestimmter Artikel	der, die, das, ein, eine, ...
CARD	Kardinalzahl	zwei [*Männer*], [*im Jahre*] 1994
FM	Fremdsprachliches Material	[*Er hat das mit* “] A big fish [” *übersetzt*]
ITJ	Interjektion	mhm, ach, tja
ORD	Ordinalzahl	[*der*] neunte [*August*]
KOUI	unterordnende Konjunktion mit “zu” und Infinitiv	um [*zu leben*], anstatt [*zu fragen*]
KOUS	unterordnende Konjunktion mit Satz	weil, dass, damit, wenn, ob
KON	nebenordnende Konjunktion	und, oder, aber
KOKOM	Vergleichskonjunktion	als, wie
NN	normales Nomen	Tisch, Herr, [*das*] Reisen
NE	Eigennamen	Hans, Hamburg, HSV
PDS	substituierendes Demonstrativpronomen	dieser, jener
PDAT	attribuierendes Demonstrativpronomen	jener [*Mensch*]
PIS	substituierendes Indefinitpronomen	keiner, viele, man, niemand
PIAT	attribuierendes Indefinit-pronomen ohne Determiner	kein [*Mensch*], irgendein [*Glas*]
PIDAT	attribuierendes Indefinit-pronomen mit Determiner	[*ein*] wenig [*Wasser*], [*die*] beiden [*Brüder*]
PPER	irreflexives Personalpronomen	ich, er, ihm, mich, dir
PPOSS	substituierendes Possessivpronomen	meins, deiner
PPOSAT	attribuierendes Possessivpronomen	mein [*Buch*], deine [*Mutter*]
PRELS	substituierendes Relativpronomen	[*der Hund* ,] der
PRELAT	attribuierendes Relativpronomen	[*der Mann* ,] dessen [*Hund*]
PRF	reflexives Personalpronomen	sich, einander, dich, mir
PWS	substituierendes Interrogativpronomen	wer, was
PWAT	attribuierendes Interrogativpronomen	welche [*Farbe*], wessen [*Hut*]
PWAV	adverbiales Interrogativ- oder Relativpronomen	warum, wo, wann, worüber, wobei
PAV	Pronominaladverb	dafür, dabei, deswegen, trotzdem
PTKZU	“zu” vor Infinitiv	zu [*gehen*]
PTKNEG	Negationspartikel	nicht
PTKVZ	abgetrennter Verbzusatz	[*er kommt*] an, [*er fährt*] rad
PTKANT	Antwortpartikel	ja, nein, danke, bitte
PTKA	Partikel bei Adjektiv oder Adverb	am [*schönsten*], zu [*schnell*]
SGML	SGML Markup	
SPELL	Buchstabierfolge	S-C-H-W-E-I-K-L
TRUNC	Kompositions-Erstglied	An- [*und Abreise*]
VVFIN	finites Verb, voll	[*du*] gehst, [*wir*] kommen [*an*]
VVIMP	Imperativ, voll	komm [*!*]
VVINF	Infinitiv, voll	gehen, ankommen
VVIZU	Infinitiv mit “zu”, voll	anzukommen, loszulassen
VVPP	Partizip Perfekt, voll	gegangen, angekommen
VAFIN	finites Verb, aux	[*du*] bist, [*wir*] werden
VAIMP	Imperativ, aux	sei [*ruhig !*]
VAINF	Infinitiv, aux	werden, sein
VAPP	Partizip Perfekt, aux	gewesen
VMFIN	finites Verb, modal	dürfen
VMINF	Infinitiv, modal	wollen
VMPP	Partizip Perfekt, modal	gekonnt, [*er hat gehen*] können
XY	Nichtwort, Sonderzeichen enthaltend	3:7, H2O, D2XW3
\$,	Komma	,
\$.	Satzbeendende Interpunktion	. ? ! ; :
\$(	sonstige Satzzeichen; satzintern	- [,]

Performance: How Difficult is POS Tagging?

How many tags can algorithms correctly predict (tag accuracy)?

- With current methods about 97.85% (https://aclweb.org/aclwiki/POS_Tagging_(State_of_the_art))

But a simple naïve baseline can reach 90% already:

- Tag every token with its most frequent tag
- Tag unknown tokens as nouns

This simple methods performs well because:

- Many words are unambiguous
- The algorithm “gets points” for unambiguous tokens, e.g.
- Articles such as `the`, `a`, etc.
- Punctuation marks

Feature-based Tagging

Word-level Features for Tag Prediction

We can obtain a lot of information from a word itself to create tagging rules:

- Word `the` `the` → `DT`
- Prefixes: `unhealthy` `un-` → `JJ`
- Suffixes: `importantly` `-ly` → `RB`
- Capitalization: `Meridian` `CAP` → `NNP`

Tag Sequence Features for Tag Prediction

Knowledge of tag combinations for neighboring words enables us to leverage syntactic rules to make decisions.

Example:

- Input: `Bill saw that man yesterday`
- Sequence 1: `NP NN DT NN NN`
- Sequence 2: **`VB VBD IN VB NN`**

Syntactically impossible sequence

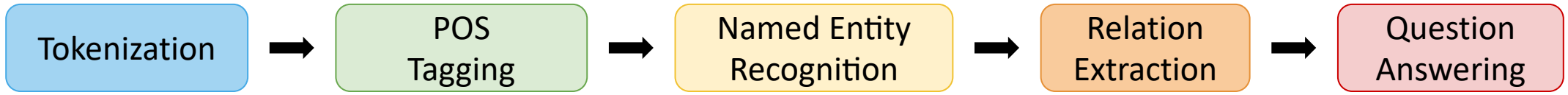
POS Performance Improvement (and Why It Matters)

Using only word-level and tag sequence features:

- We can reach 93.7% tag accuracy already
- "Woah, we're **half way** there"
  (on the way to state-of-the-art performance)

So why should we care about the remaining 6.3%?

- NLP happens in what we call "The NLP Pipeline"
- Error propagation is a critical issue



Towards More Advanced POS Tagging

Probabilistic methods

- Use a **corpus for training**
- Obtain POS probabilities from **manual annotations** of the corpus
- Language-agnostic: Method can be applied to different languages

Rule-based methods

- Use linguistic knowledge to solve ambiguous cases
- Knowledge is mapped to rules
- Rules are easy to interpret
- Each language needs its own rules

Probabilistic Tagging

Tagging With Lexical Frequencies

Problem: How can we determine the most likely POS tag for a given token?
Solution: Derive probabilities from a large, annotated text corpus

- Count POS tag frequencies for a given token in the annotated corpus
- Select the most probable (= frequent) POS tag in new, unseen data

Tagging With Lexical Frequencies

Example:

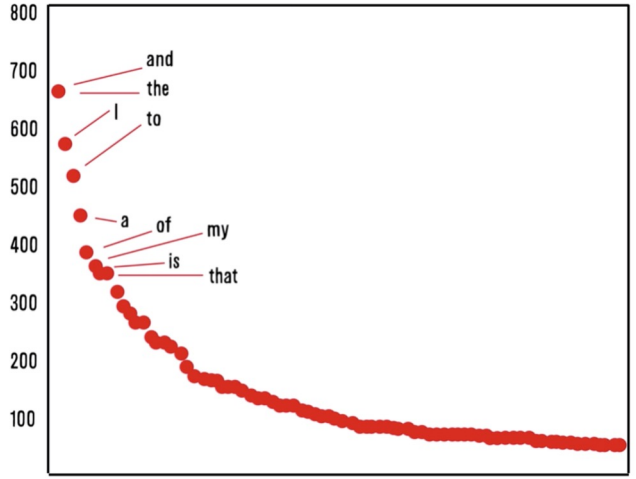
- `Sebastian[NNP] Vettel[NNP] is[VBZ] expected[VBN] to[TO] race[VB] tomorrow[NN].`
- `People[NNS] continue[VBP] to[TO] inquire[VB] the[DT] reason[NN] for[IN] the[DT] race[NN] to[IN] outer[JJ] space[NN].`
- To assign a tag to `race`, given its lexical frequency, we choose the tag that has the higher probability: **p(VB|race)** or **p(NN|race)**?

Unigram Tagger: A Naïve Prediction Model

- How a unigram tagger works:
- We have access to an **annotated text corpus** (= our training data)
- We create **statistics** of how many times each token is seen with each POS tag (= the training process)
- Based on these learned frequencies, we use the trained model on new, **unseen text** to associate with each word its most likely POS tag (= prediction)

Is it really this easy?

- Unfortunately, no!
- **Problem**: many words might never be seen in the training data (out-of-vocabulary)
- **Solution**: specify a default tag as “backoff”



Unigram Tagger Example: Training

Training data:

```
Consider that result that George has achieved. That was
simply amazing, although, he did not learn anything.
```

Manual tagging of the training data produces:

```
Consider[VB] that[DT] result[NN] that[PRP] George[NNP]
has[VB] achieved[VB] .[SYM] That[PRP] was[VB] simply[RB]
amazing[RB] ,[SYM] although[CC] ,[SYM] he[PRP] did[VB]
not[RB] learn[VB] anything[NN] .[SYM]
```

(For simplicity, all verb forms are collapsed to VB in the example)

Unigram Tagger Example: Prediction

Input test sentence:
`George achieved that result.`

Tagging of input according to the training data:
`George[NNP] achieved[VB] **that[k]** result[NN] .[SYM]`

Prediction:

- `George`, `achieved`, `result`, and `.` are unambiguous and can be assigned a POS tag directly
- `k = [DT] | [PRP]` requires a decision based on the training data

Unigram Tagger Example: Prediction Probabilities

Manually tagged training data:

```
Consider[VB] that[DT] result[NN] that[PRP] George[NNP]
has[VB] achieved[VB] .[SYM] That[PRP] was[VB] simply[RB]
amazing[RB] ,[SYM] although[CC] ,[SYM] he[PRP] did[VB]
not[RB] learn[VB] anything[NN] .[SYM]
```

Test input:

```
George[NNP] achieved[VB] that[k] result[NN] .[SYM]
```

POS probabilities according to the model trained on the training data:

```
p(k=DT|that) = 1/3
p(k=PRP|that) = 2/3
```

→ the model predicts that[PRP]

Unfortunately,
[PRP] is wrong!

Unigram Tagger: The Fundamental Problems

- Why is the unigram tagger not working?
- We are not computing probabilities properly: use Bayes' rule instead!
  Among all possible tags `k'`, the best choice for `k=k'` maximizes:

$$p(k = k'|that) = \frac{p(that|k = k') * p(k')}{p(that)}$$

$p(that)$ ← constant (can be ignored)

But, more simply: The most frequent tag is obviously not always correct!

- We need to know in which sense `that` is being used.

→ We need to take the **context** into account.

- `George[NNP] achieved[VB]` **`that[k] result[NN]`** `.[SYM]`

N-grams: Encoding Cooccurrence Context

An **n-gram** is a contiguous sequence of n items from a given sample of text.

- The n stands for how many terms are used
- `n = 1`: **unigram**, one token (0th order)
- `n = 2`: **bigram**, two tokens (1st order)
- `n = 3`: **trigrams**, three tokens (2nd order)
- etc., but n > 5 is rare (take a minute to figure out why)

Example:

- Input: `George[NNP] achieved[VB] that[DT] result[NN]`
- Token bigrams: George achieved; achieved that; that result
- Tag bigrams: `NNP VB; VB DT; DT NN`

Using N-grams

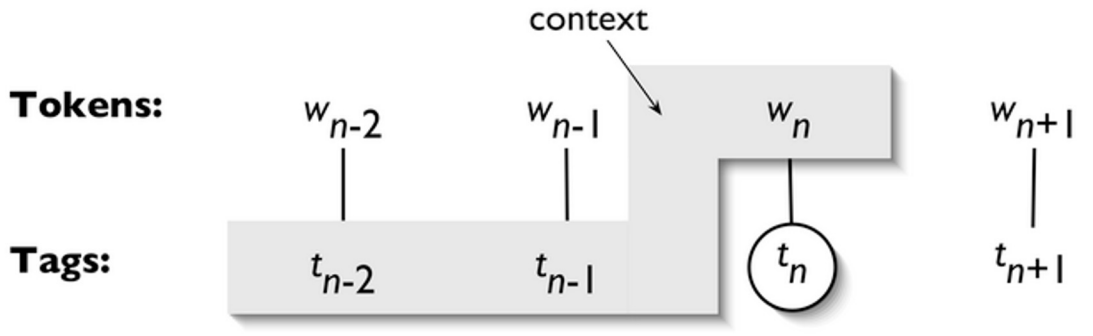
N-grams can encode different types of contexts, e.g.,

- Character based n-grams
- Token-based n-grams
- POS-based n-grams
- Combinations of the above

Ordering and context in n-grams:

- We often define context to mean “adjacent”, but this is not required
- We use n-grams to help us determine the context in which some linguistic phenomenon happens, e.g.,
  - Words before and after a period to determine if it is the end of a sentence
  - Detecting **open compound nouns** (e.g., `outer space`)

N-gram POS tagger



Core idea:

- Use the unigram probability for the current token (→ unigram tagger)
- Use the predicted preceding n-1 **tags**
- Optionally: We could also use the preceding n-1 **tokens**, but this is impractical

Bigram tagger:

- Prediction: What is the most likely tag for token `n`, given token `n` and tag `n-1`?
- The tagger picks the tag which is most likely, given that context

Bigram Tagger Example

Back to our manually tagged training data:

```
Consider[VB] that[DT] result[NN] that[PRP] George[NNP]
has[VB] achieved[VB] .[SYM] That[PRP] was[VB] simply[RB]
amazing[RB] ,[SYM] although[CC] ,[SYM] he[PRP] did[VB]
not[RB] learn[VB] anything[NN] .[SYM]
```

Test input:

```
George[NNP] achieved[VB] that[k] result[NN] .[SYM]
```

Preceding the token that is a token with the tag [VB]. Thus, we obtain:

```
p(k=DT|that) = p(k=DT|VB)  * p(that|DT)  = 1/6 * 1/1 = 1/6
p(k=PRP|that)= p(k=PRP|VB) * p(that|PRP) = 0/6 * 2/3 = 0
```

→ the model selects `that[DT]` as prediction ✓

Combining Multiple Taggers

To improve tagging accuracy, we should use more accurate algorithms when we can and a simpler algorithms with broader coverage when needed.

- Attempt to tag the token with the 1st order tagger (e.g., bigram tagger)
- If the 1st order tagger is unable to find a tag for the token, fall back to using a 0th order tagger (e.g., unigram tagger)
- If the 0th order tagger is also unable to find a tag, use the default tagger to find a tag (e.g., feature based tagger)

Important note:
Bigram and trigram taggers need the previous tag context to assign new tags. If they see a `[na]` tag in the previous context, they will also output `[na]`.
→ Error propagation

Rule-based Tagging

Rule-based Tagging

Valid reasons for linguistic complaints? Yes!

- Languages are more than cooccurrences statistics!
- This is just a heap of numbers and probabilities!
- Where is the linguistic knowledge?

Solution:

- We can use **handcrafted sets of rules** to tag input sentences
- For example, if a token follows a determiner, tag it as a noun

The Brill Tagger

An example of Transformation-based Learning

- Core idea:
  - Use a probabilistic approach first
  - Then revise and correct it using contextual rules
- Similar to painting: sketch first, then paint properly
- Very popular (freely available and works fairly well)
- The rules are linguistically interpretable
- It is a supervised method and therefore requires a tagged corpus
- Example: http://cst.dk/online/pos_tagger/uk/index.html

Brill Tagger: Process

Initialization:

- Tag each word with the most likely POS (as observed in the corpus)
- For words that are not in the corpus:
  - If it is capitalized: label as noun
  - For tokens that end in the same three letters as a word with known tag: use the tag of the known word (e.g., `label`**`ing`**`[VBG]`)

Patching:

- Find suitable instances and apply patches to improve the result
- Example of a patch:
  - If one of the two preceding words is tagged as a determiner, change the tag of tokens that are tagged as a verb to noun

Brill Tagger: Example

Input:	Ask	congress	to	increase	grants.
Gold standard:	VB	NP	TO	VB	NNS
Unigram tagger:	VB	NP	TO	**NN**	NNS

Patch: Replace `[NN]` with `[VB]` when the previous tag is `[TO]`

Brill Tagger: Patch Templates

- The preceding (following) token is tagged `Z`
- The token two before (after) is tagged `Z`
- One of the two preceding (following) tokens is tagged `Z`
- One of the three preceding (following) tokens is tagged `Z`
- The preceding (following) word is tagged `Z` and the following token is tagged `W`
- The preceding (following) word is tagged `Z`
  and the word two before (after) is tagged `W`
- The current word is (is not) capitalized
- The previous word is (is not) capitalized

Brill Tagger: Patches

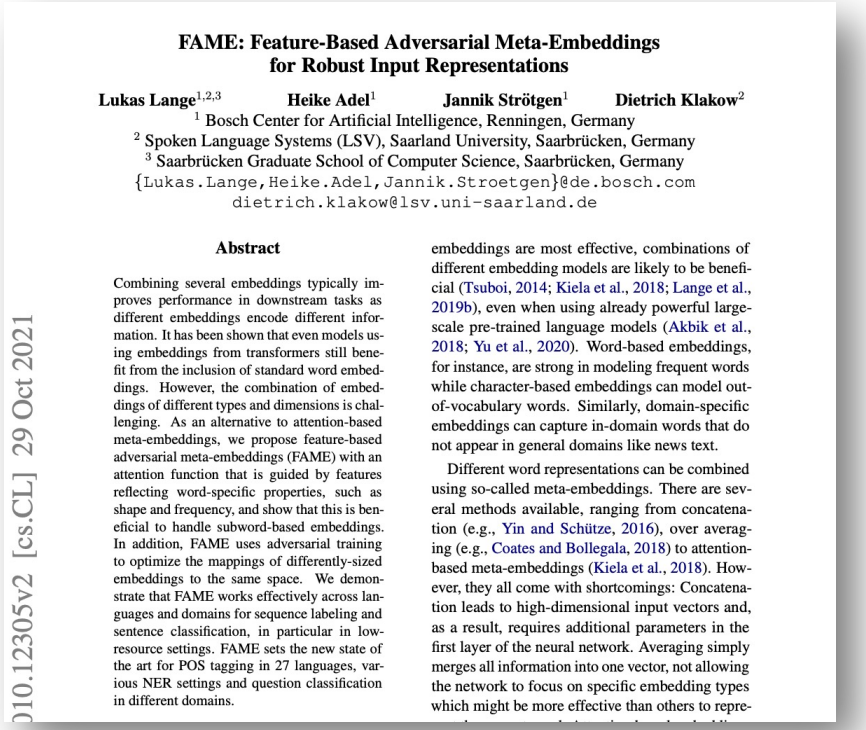
To generate patching rules:

- Compare initial tags with correct tags, setup a list of tagging error triples: (correct tag, incorrect tag, frequency)
- For each error triple, find the patch template whose application results in the greatest error reduction.
- For each error triple, compute the reduction in error that results from applying the patch (subtracting the number of new errors...).
- Add the patch that results in the greatest improvement to the list of patches and apply it to the corpus.
- Repeat until no further improvement can be gained.

→ Result: tagging procedure (ordered list of transformations), which can be applied to new, untagged text.

POS Tagging – Today (Example)

- Neural network-based methods are common for pos tagging with sota performance
- Either Bi-LSTMs or transformers can be used



FAME: Feature-Based Adversarial Meta-Embeddings for Robust Input Representations

Lukas Lange[1,2,3] **Heike Adel**[1] **Jannik Strötgen**[1] **Dietrich Klakow**[2]
[1] Bosch Center for Artificial Intelligence, Renningen, Germany
[2] Spoken Language Systems (LSV), Saarland University, Saarbrücken, Germany
[3] Saarbrücken Graduate School of Computer Science, Saarbrücken, Germany
{Lukas.Lange,Heike.Adel,Jannik.Stroetgen}@de.bosch.com
dietrich.klakow@lsv.uni-saarland.de

Abstract

Combining several embeddings typically improves performance in downstream tasks as different embeddings encode different information. It has been shown that even models using embeddings from transformers still benefit from the inclusion of standard word embeddings. However, the combination of embeddings of different types and dimensions is challenging. As an alternative to attention-based meta-embeddings, we propose feature-based adversarial meta-embeddings (FAME) with an attention function that is guided by features reflecting word-specific properties, such as shape and frequency, and show that this is beneficial to handle subword-based embeddings. In addition, FAME uses adversarial training to optimize the mappings of differently-sized embeddings to the same space. We demonstrate that FAME works effectively across languages and domains for sequence labeling and sentence classification, in particular in low-resource settings. FAME sets the new state of the art for POS tagging in 27 languages, various NER settings and question classification in different domains.

embeddings are most effective, combinations of different embedding models are likely to be beneficial (Tsuboi, 2014; Kiela et al., 2018; Lange et al., 2019b), even when using already powerful large-scale pre-trained language models (Akbik et al., 2018; Yu et al., 2020). Word-based embeddings, for instance, are strong in modeling frequent words while character-based embeddings can model out-of-vocabulary words. Similarly, domain-specific embeddings can capture in-domain words that do not appear in general domains like news text.

Different word representations can be combined using so-called meta-embeddings. There are several methods available, ranging from concatenation (e.g., Yin and Schütze, 2016), over averaging (e.g., Coates and Bollegala, 2018) to attention-based meta-embeddings (Kiela et al., 2018). However, they all come with shortcomings: Concatenation leads to high-dimensional input vectors and, as a result, requires additional parameters in the first layer of the neural network. Averaging simply merges all information into one vector, not allowing the network to focus on specific embedding types which might be more effective than others to repre-

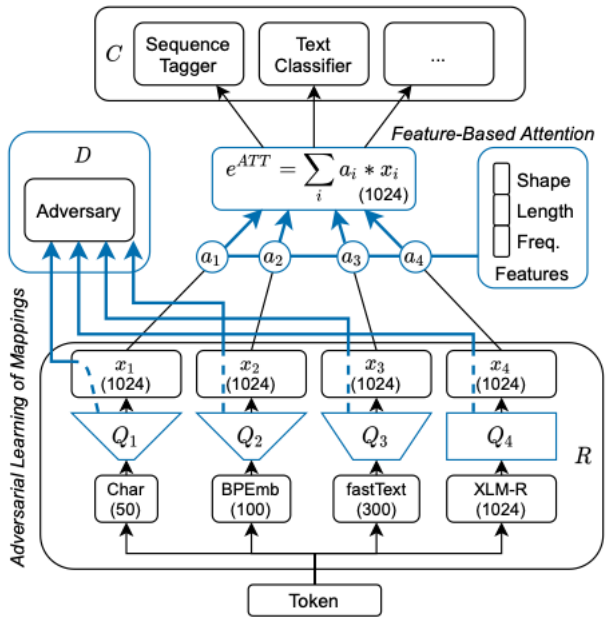


Figure 1: Overview of the FAME model architecture. Blue lines highlight our contributions. C (classifier), D (discriminator) and R (input representation) denote the components of adversarial training. The dimensions of intermediate representations are given in parentheses.

POS Tagging – Today (Example)

- POS tagging results on high and low resource languages are quite good nowadays.

- In our work, good input representation was the key!
- We will learn about “input representation” and “embeddings” in later lectures.

	SOTA1	SOTA2	SOTA3	FAME
Bg (Bulgarian)	97.97	98.53	98.7	**99.53**
Cs (Czech)	98.24	98.81	98.9	**99.33**
Da (Danish)	96.35	96.74	97.0	**99.13**
De (German)	93.38	94.35	94.0	**95.95**
En (English)	95.17	95.82	95.6	**98.09**
Es (Spanish)	95.74	96.44	96.5	**97.75**
Eu (Basque)	95.51	94.71	95.6	**97.66**
Fa (Persian)	97.49	97.51	97.1	**98.68**
Fi (Finnish)	95.85	95.40	94.6	**98.67**
Fr (French)	96.11	96.63	96.2	**97.19**
He (Hebrew)	96.96	97.43	96.6	**98.00**
Hi (Hindi)	97.10	97.21	97.0	**98.35**
Hr (Croatian)	96.82	96.32	96.8	**97.96**
Id (Indonesian)	93.41	94.03	93.4	**94.24**
It (Italian)	97.95	98.08	98.1	**98.82**
Nl (Dutch)	93.30	93.09	93.8	**94.74**
No (Norwegian)	98.03	98.08	98.1	**99.16**
Pl (Polish)	97.62	97.57	97.5	**99.05**
Pt (Portuguese)	97.90	98.07	98.2	**98.86**
Sl (Slovenian)	96.84	98.11	98.0	**99.44**
Sv (Swedish)	96.69	96.70	97.3	**99.17**
Avg.	96.40	96.65	96.6	**98.08**
El (Greek)	-	98.24	97.9	**98.89**
Et (Estonian)	-	91.32	92.8	**97.07**
Ga (Irish)	-	91.11	91.0	**94.27**
Hu (Hungarian)	-	94.02	94.0	**97.72**
Ro (Romanian)	-	91.46	89.7	**96.64**
Ta (Tamil)	-	83.16	88.7	**91.10**
Avg.	-	91.55	92.4	**95.95**

Table 4: POS tagging results (accuracy) (using gold segmentation). SOTA1 refers to results from Plank et al. (2016), SOTA2 to Yasunaga et al. (2018) and SOTA3 to Heinzerling and Strube (2019). As Yasunaga et al. (2018), we split into high-resource (top) and low-resource languages (bottom).

Further-Watching Material

Introduction to N-grams

https://youtu.be/O7k8M8FwGLg?list=PLoROMvodv4rOFZnDyrlW3-nI7tMLtmiJZ

Estimating N-gram Probabilities

https://youtu.be/JfOdKOmYtqI?list=PLoROMvodv4rOFZnDyrlW3-nI7tMLtmiJZ

Thank you for your attention!

Questions?