University of Zurich[UZH]

Institute of Computational Linguistics

Machine Translation

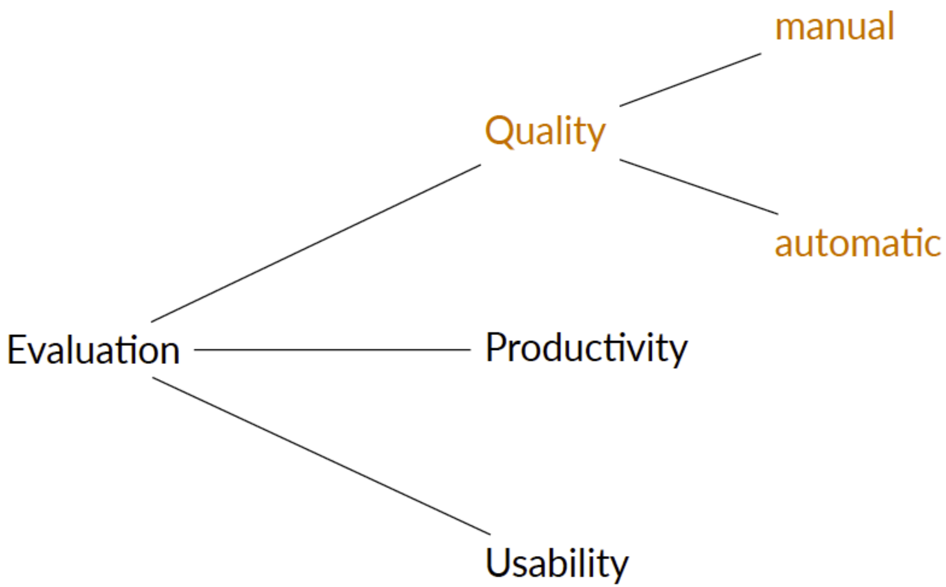
3 Preprocessing

+ postprocessing

Mathias Müller

Last week

How to evaluate quality?



Termin	Thema
19.02.	Einführung; regelbasierte vs. datengetriebene Modelle
26.02.	Evaluation
05.03.	Trainingsdaten, Vor- und Nachverarbeitung
12.03.	N-Gramm-Sprachmodelle, statistische Maschinelle Übersetzung
19.03.	Grundlagen Lineare Algebra und Analysis, Numpy
26.03.	Lineare Modelle: lineare Regression, logistische Regression
02.04.	Neuronale Netzwerke: MLPs, Backpropagation, Gradient Descent
09.04.	Word Embeddings, Recurrent neural networks
16.04.	Tensorflow und Google Cloud Platform
30.04.	Encoder-Decoder-Modell
07.05.	Decoding-Strategien
14.05.	Attention-Mechanismus, bidirektionales Encoding, Byte Pair Encoding
21.05.	Maschinelle Übersetzung in der Praxis (Anwendungen)
28.05.	Zusammenfassung, Q&A Prüfung
Eventuell: Gastvortrag Prof. Artem Sokolov	
04.06., Raum TBA, 16:15 bis 18:00 Uhr	
Prüfung (schriftlich)	
18.06., AND-2-48, 16.15 bis 18:00 Uhr	

EVALUATION

TRAINING DATA

SMT

NMT

this is kinda important

Topics of today

- training data for machine translation systems
- learn how text is processed
  - before training and translation (**preprocessing**)
  - after translation (**postprocessing**)

Training data

The European Community had already agreed to phase out CFCs by 1997 and hoped that other countries would do the same.

La Comunidad Europea ya había convenido en suprimir los CFC para 1997 y confiaba en que otros países hicieran lo mismo.

The Protocol should be amended to reflect that situation.

El Protocolo debía enmendarse para reflejar esa situación.

But that was not enough.

No obstante, eso no bastaba.

The Technology and Economics Assessment Panel should be asked to assess the implications of phasing out halons, carbon tetrachloride and methyl chloroform also by 1997.

Se debería pedir al Grupo de evaluación técnica y económica que evaluara las repercusiones de la supresión gradual de los halones, el tetracloruro de carbono y el metilcloroformo también para 1997.

Where do people get data?

- researchers: freely available corpora - some even need to be translated & public by law

Europarl, UN, JRC, opensubtitles

- companies: may have proprietary data, or crawl the web

Autodesk, Google, Amazon

How much data?

- Rule of thumb: 10m (million) sentence pairs for "reasonable" performance

- commercial, general-domain systems have > 100m sentence pairs

Google Translate

total = 10 m

Splitting data into three parts

train

9'996'000

development/
validation/
dev

2000

test

2000

SMT and NMT systems use training data

```python
class TranslationSystem:

    def train(self, source_sentences, target_sentences):
        # estimate probabilities from training data

    def translate(self, source_sentence):
        # pick most probable translation
        return target_sentence

source_sentences = open("train.de").readlines()
target_sentences = open("train.en").readlines()

ts = TranslationSystem()
ts.train(source_sentences, target_sentences)
```

But only after preprocessing!

Train only after preprocessing

```
class TranslationSystem:

    def train(self, source_sentences, target_sentences):
        # estimate probabilities from training data

    def translate(self, source_sentence):
        # pick most probable translation
        return target_sentence

preprocessed_source_sentences = []
preprocessed_target_sentences = []

for source_sentence in open("train.de"):
    preprocessed_source_sentences.append(preprocess_sentence(source_sentence))
for target_sentence in open("train.en"):
    preprocessed_target_sentences.append(preprocess_sentence(target_sentence))

ts = TranslationSystem()
ts.train(preprocessed_source_sentences, preprocessed_target_sentences)
```

Translation: only after preprocessing (+ postprocessing!)

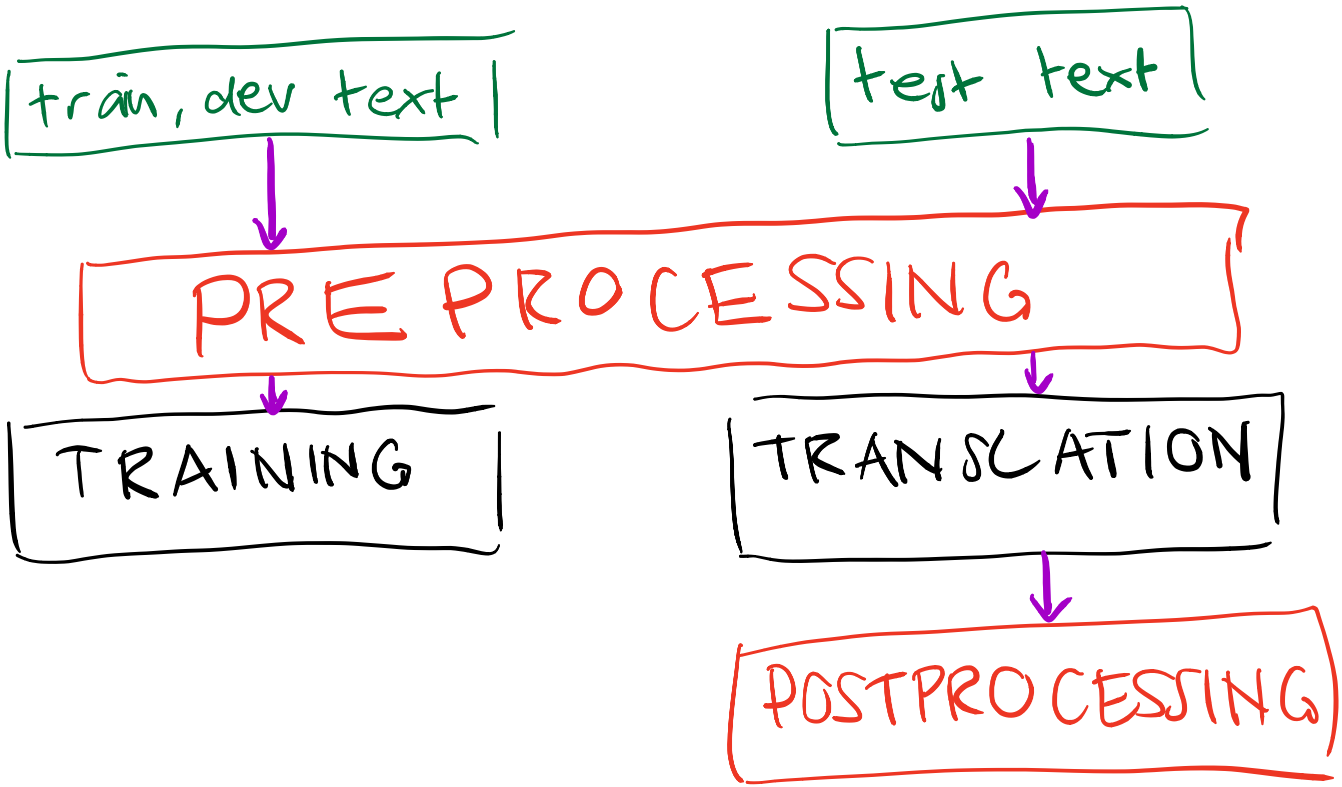
```
ts = TranslationSystem()
ts.train(source_sentences, target_sentences)

ts.translate("This is a test sentence")
```

ohne pre- oder postprocessing

```
ts = TranslationSystem()
ts.train(preprocessed_source_sentences, preprocessed_target_sentences)

ts.translate(preprocess_sentence("This is a test sentence"))
```

```
ts = TranslationSystem()
ts.train(preprocessed_source_sentences, preprocessed_target_sentences)

translation = ts.translate(preprocess_sentence("This is a test sentence"))

translation = postprocess_sentence(translation)
```

Pre- and postprocessing in pictures



Typical preprocessing steps (order important)

- normalization
- tokenization
- a method to improve casing

```python
def preprocess_sentence(sentence):
    sentence = normalize(sentence)
    sentence = tokenize(sentence)
    sentence = truecase(sentence)

    return sentence
```

Normalization

- lump together different unicode characters that are similar
- remove non-printing characters
- check encoding

' ’

'

Example for a normalization function

```python
_emoji_pattern = re.compile(
    u"(\ud83d[\ude00-\ude4f])|"  # emoticons
    u"(\ud83c[\udf00-\uffff])|"  # symbols & pictographs (1 of 2)
    u"(\ud83d[\u0000-\uddff])|"  # symbols & pictographs (2 of 2)
    u"(\ud83d[\ude80-\udeff])|"  # transport & map symbols
    u"(\ud83c[\udde0-\uddff])"  # flags (iOS)
    "+", flags=re.UNICODE)


def normalize(string_):
    """
    Remove most emojis from text.
    """
    return _emoji_pattern.sub(r'', string_)
```

2) Tokenization

- purpose: make sure that whitespaces can be interpreted as token boundaries
- note: in MT numbers, punctuation etc. are tokens, too and are not removed! **WHY?**

"Haus!" → "Haus !"

whitespace

Example for tokenization function

```python
def tokenize(string_, verbose=False):
    """
    Scan a string by iteratively matching regexes.
    Source:
    https://stackoverflow.com/a/693818/1987598

    :param string_: untokenized input string
    :param verbose: whether the type of tokens should be returned
    """

    scanner = re.Scanner([
        (r"[0-9]+", lambda scanner, token:("NUMERIC", token)),
        (r"\w+", lambda scanner, token:("WORD", token)),
        (r"[,.!?/]+", lambda scanner, token:("PUNCTUATION", token)),
        (r"\s+", None),  # None == skip token.
    ])

    scan, _ = scanner.scan(string_)

    if verbose:
        return scan
    else:
        return [token[1] for token in scan]
```

2) Tokenize an example sentence

Hausnummer

my script:

```
sysadmins-MBP:3 mathiasmuller$ echo "Is Dr. Phil rich etc.?" | python tokenize.py
Is Dr . Phil rich etc .?
```

Moses tokenizer:

```
mmueller@vigrid ~/me@karr/robustness_experiments/scripts
 % echo "Is Dr. Phil rich etc.?" | $MOSES_HOME/scripts/tokenizer/tokenizer.perl -l en -q
Is Dr. Phil rich etc . ?
```

Dr. ✓ etc . ✗

3)
Casing

Purpose of casing: make sure case of characters in words is consistent

Haus haus HAus hAus

Known casing methods are

- **truecasing** most frequent casing
- recasing 2nd translation system for casing
- selfcasing

Truecasing

- idea: reduce all words to their most frequent case
- trains a **truecasing model** on training data   Haus : 17361   HaUs : 37
- training means: count how often each word occurs (simple!)

Example for truecase function

```python
def build_model(corpus, verbose=False):

    words = re.findall(r'\w+', corpus)
    counter = Counter(words)

    if verbose:
        sys.stderr.write(repr(counter) + "\n")
    return counter


def truecase(line, model):

    truecased = []
    for token in line.split():
        count = model[token]
        lowercase_count = model[token.lower()]

        if count > lowercase_count:
            truecased.append(token)
        else:
            truecased.append(token.lower())

    return " ".join(truecased)
```

Truecase an example sentence

```
sysadmins-MBP:3 mathiasmuller$ echo 'sudden shower of god God god' | python truecase.py
Counter({u'god': 2, u'shower': 1, u'of': 1, u'sudden': 1, u'God': 1})
sudden shower of god god god
```

Summary Preprocessing

- several processing steps are applied before training and before translation
- typical steps are:
  - normalization
  - tokenization
  - casing (most common: truecasing)
- steps must be **identical** for training and translation

Haus.

↳ Haus .

Postprocessing steps

- which steps and order depends on preprocessing:
- postprocessing **undoes** preprocessing steps

PRE

normalize
tokenize
true casing

POST

detokenization
detruecasing

Restore original casing

e f

- for truecasing very simple; for Western languages typically uppercase first letter in a sentence

"Was für ein HAus!"

tok was für ein HAus ! "

tru was für ein Haus !

what a house !

detru What a house !

Detokenization

- Remove unnatural whitespace

"Was für ein HAus!"

tok " was für ein HAus ! "

tru was für ein Haus !

what a house !

detru What a house !

detok What a house!

Summary Postprocessing steps

- postprocessing undoes preprocessing steps, in reverse order

Depends heavily on languages involved!

- actual processing steps depend on the languages
- examples: some languages do not use whitespace, languages have different alphabets

~~tokenization~~ segmentation

transliteration BPE

Languages like Thai need segmentation

ดีวันก่อน ได้มีเวลาไป update webpage ของตัวเอง แล้วก็ได้เพิ่ม webstat เข้า
วามจริงของเดิมก็มีอยู่แล้ว แต่เป็น ของ truehits ซึ่งตอนนี้เหมือนกันว่าไม่ฟรีแล้
ยต้องเปลี่ยนตัวใหม่) พอผ่านไปไม่กี่วันก็ลองเข้าไปดูสถิติ ก็มีคนเข้าไปเยี่ยมเว็บ
งเรา เพราะว่าสนใจเรื่อง โปรแกรมตัดคำ SWATH ประกอบกับตอนนี้ก็ว่าจะกลั
ทำวิจัย เรื่องตัดคำต่อ หลังจากห่างหายไปนาน เพราะมัวแต่ไปทำด้าน speech
cognition กับ machine tran
นว่าเดี๋ยวจะเขียนเรื่องตัดคำก
ก็เสนอแนะเข้ามานะครับ เพ
ะโยชน์กันเยอะๆ

Tom Hoar
Re: [Moses-support] handling no-space languages in the decoder
To: moses-support@mit.edu

19 December 2017 at 09:35

TH

Hi Ryan,

Mathias is correct about preprocessing and not modifying the SMT model. His suspicion that removing spaces is even less of a problem" is not the case.

Is spaces were almost never used in Thai (like in Chinese for example), removing them would be trivial. However, Thai has over 50 "proper" uses for spaces, but word delimiter is not one. Add to that the disinclination for Thais to follow the rules (depending on the text type), and restoring proper spacing becomes a quite complex task.

Practial tips

- do not reinvent the wheel, there are standard implementations of pre- and postprocessing

- save and look at intermediate steps to debug

```
cat train.de | normalize.py | tokenize.py |
truecase.py > train.tc.de
```

Summary

SMT 1m NMT 10m

- MT systems need at least on the order of 10m sentence pairs to perform well
- at the periphery of training and translation
  - text is preprocessed before training and translation
  - text is postprocessed after translation

Next time: statistical machine translation (SMT)

Termin	Thema
19.02.	Einführung; regelbasierte vs. datengetriebene Modelle
26.02.	Evaluation
05.03.	Trainingsdaten, Vor- und Nachverarbeitung
12.03.	N-Gramm-Sprachmodelle, statistische Maschinelle Übersetzung
19.03.	Grundlagen Lineare Algebra und Analysis, Numpy
26.03.	Lineare Modelle: lineare Regression, logistische Regression
02.04.	Neuronale Netzwerke: MLPs, Backpropagation, Gradient Descent
09.04.	Word Embeddings, Recurrent neural networks
16.04.	Tensorflow und Google Cloud Platform
30.04.	Encoder-Decoder-Modell
07.05.	Decoding-Strategien
14.05.	Attention-Mechanismus, bidirektionales Encoding, Byte Pair Encoding
21.05.	Maschinelle Übersetzung in der Praxis (Anwendungen)
28.05.	Zusammenfassung, Q&A Prüfung
Eventuell: Gastvortrag Prof. Artem Sokolov	
04.06., Raum TBA, 16:15 bis 18:00 Uhr	
Prüfung (schriftlich)	
18.06., AND-2-48, 16.15 bis 18:00 Uhr	

EVALUATION

TRAINING DATA

SMT

NMT

this is kinda important