IMS tutorial

Yixing Luan

What is IMS ?

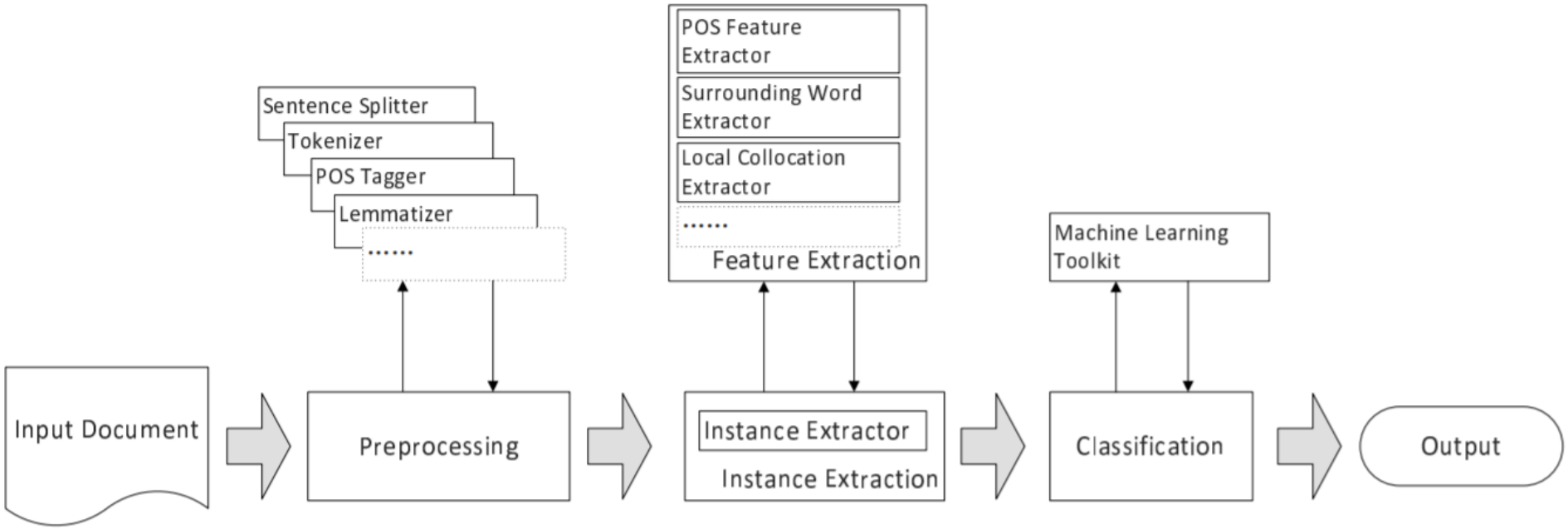
- “It Makes Sense: A Wide-Coverage Word Sense Disambiguation System for Free Text (Zhong and Ng, ACL2010)”

- SVM based WSD system

  - train word-expert models
    → different models for each word type (lemma, pos)
    → classes are senses for the word type

  - use 3 conventional WSD features
    - Surrounding words
    - POS tags of Surrounding words
    - Local collocations

What is IMS ?



* Figure 1: IMS system architecture

* Taken from “It Makes Sense: A Wide-Coverage Word Sense Disambiguation System for Free Text (Zhong and Ng, ACL2010)” page 80

Feature Expression

- Bag-of-Words (BoW)
  - Have a feature for each word in some subset of the vocabulary
  - Usually function words are excluded
  - Limited to words in some context window

Training: The **bank adjusts** the **interest rates** that it **takes** …
The **manager said** an **interest rate** is the **percentage** of …
… etc.

Feature(**interest**) = {…, **bank**, **adjust**, **rate**, **take**, **manager**, **say**, **percentage**, …}
→ {**0**, …, **0**, **0**, **0**, **0**, **0**, **0**, **0**, …, **0** }

Testing: If those words appear in the test context, change the dimension 0 → 1

Feature 1: Surrounding Words (BoW)

- Ignore function words
- Lower-cased lemma
- Relative position is ignored

Training: The **bank adjusts** the **interest** **rates** that it **takes** …

Feature(**interest**) = {…, **bank**, **adjust**, **rate**, **take**,…}

→ {…, **0** , **0** , **0** , **0** ,…}

Testing: My brother always **takes** an **interest** in my work .

Feature(**interest**) = {…, **0**, **0**, **0**, **1**,…}

Feature 2: POS tags of surrounding words

- 3 words to the left and to the right + the target word

RB VBZ DT NN IN PPR$ NN
My brother always takes an **interest** in my work .

Feature(**interest**) = {**RB, VBZ, DT, NN, IN, PPR$, NN**}

- Relative position is ignored

Feature 3: Local Collocations

- 11 local collocations → Each collocation forms feature vector

  My brother always takes an **interest** in my work .

  - C(-2, -2) = takes
  - C(-1, -1) = an
  - C(1, 1) = in
  - C(2, 2) = my
  - C(-2, -1) = takes an
  - C(-1, 1) = an interest in
  - C(1, 2) = in my
  - C(-3, -1) = always takes an
  - C(-2, 1) = takes an interest in
  - C(-1, 2) = an interest in my
  - C(1, 3) = in my work

- Pay attention to relative position

IMS + embeddings

- “Embeddings for Word Sense Disambiguation: An Evaluation Study (Iacobacci et al., ACL2016)”

- Integrate 400-d skip-gram Word2Vec embeddings as a feature

- IMS with all features vs IMS without the surrounding word feature
  - original IMS surrounding word feature
  - Word2Vec feature

Performance

*

	Tr. Corpus	System	Senseval-2	Senseval-3	SemEval-07	SemEval-13	SemEval-15
Supervised	**SemCor**	IMS	70.9	69.3	61.3	65.3	69.5
		IMS+emb	71.0	69.3	60.9	**67.3**	71.3
		IMS$_{-s}$+emb	**72.2**	**70.4**	**62.6**	65.9	71.5
		Context2Vec	71.8	69.1	61.3	65.6	**71.9**
		MFS	65.6	66.0	54.5	63.8	67.1
		Ceiling	*91.0*	*94.5*	*93.8*	*88.6*	*90.4*
	SemCor + OMSTI	IMS	72.8	69.2	60.0	65.0	69.3
		IMS+emb	70.8	68.9	58.5	66.3	69.7
		IMS$_{-s}$+emb	**73.3**	**69.6**	61.1	66.7	70.4
		Context2Vec	72.3	68.2	**61.5**	**67.2**	**71.7**
		MFS	66.5	60.4	52.3	62.6	64.2
		Ceiling	*91.5*	*94.9*	*94.7*	*89.6*	*91.1*

* Taken from “Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison (Raganato et al., EACL2017)” Table 4

How to use IMS?

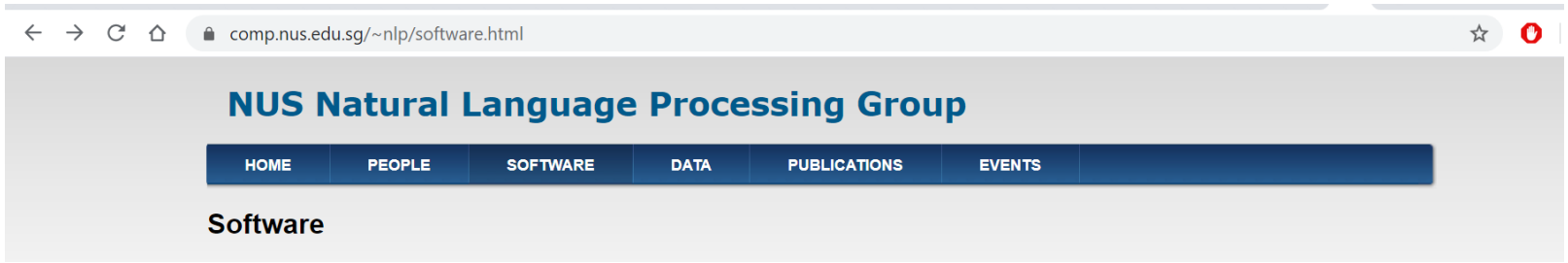
- Requirement: Java 6 or higher

- Download the IMS package from NUS homepage

- Follow README
  - Train on a sense-annotated corpus
  - Test on a WSD test sets

Installation

- Go to https://www.comp.nus.edu.sg/~nlp/software.html
- Install and uncompress packages



IMS

IMS (It Makes Sense) is a supervised English all-words word sense disambiguation (WSD) system. The flexible framework of IMS allows users to integrate different preprocessing tools, additional features, and different classifiers. By default, we use linear support vector machines as the classifier with multiple features. This implementation of IMS achieves state-of-the-art results on several SensEval and SemEval tasks. You can try the **demo** and download the software:

- **IMS - README**
- **IMS - IMS_v0.9.2**
- **IMS - IMS_v0.9.2.1**
- **IMS - lib**
- **IMS - models**
- **IMS - examples**

What's in IMS package?

- 5 bash scripts to train and test WSD models

  - train_one.bash
    - train word-expert models using a sense-annotated corpus
  - test_one.bash
    - test one lexical sample WSD using the trained models
  - testFine.bash
    - test fine-grained all-words WSD using the trained models
  - testCoarse.bash
    - test coarse-grained all-words WSD using the trained models
  - testPlain.bash
    - disambiguate a plain text using the trained models

Datasets for English All-Words WSD

- Training Data
  - SemCor: manually sense-annotated corpus
    the most popular training data for WSD
    people use the unified version* by (Raganato et al., EACL2017)

  - OMSTI: automatically sense-annotated corpus

- Test Data: 5 datasets from previous shared tasks
  - Senseval2, Senseval3, SemEval2007, SemEval2013, SemEval2015
  - Instead of the original datasets, the unified version* is used for evaluation

* "Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison (Raganato et al., EACL2017)"

Data Preprocessing (xml data)

- XML format in the unified framework (Raganato et al., EACL2017)

```
<?xml version="1.0" encoding="UTF-8" ?>
<corpus lang="en" source="semcor">
<text id="d000" source="br-e30">
<sentence id="d000.s000">
<wf lemma="how" pos="ADV">How</wf>
<instance id="d000.s000.t000" lemma="long" pos="ADJ">long</instance>
<wf lemma="have" pos="VERB">has</wf>
```

Convert
by yourself!

- XML format required by IMS

```
<?xml version="1.0" encoding="UTF-8"?>
<corpus lang="english">
    <lexelt item="long.ADJ" pos="ADJ">
        <instance docsrc="br-e30" id="d000.s000.t000">
            <context>
                How
                <head>long</head>
                has it been since you reviewed the objectives of your benefit and service program ?
            </context>
        </instance>
    </lexelt>
```

Data Preprocessing (gold key data)

- Key format in the unified framework (Raganato et al., EACL2017)
  - instance_id <space> gold_key

```
d000.s000.t000 long%3:00:02::
d000.s000.t001 be%2:42:03::
d000.s000.t002 review%2:31:00::
d000.s000.t003 objective%1:09:00::
```

- XML format required by IMS
  - lexical_item <space> instance_id <space> gold_key

```
long.ADJ d000.s000.t000 long%3:00:02::
be.VERB d000.s000.t001 be%2:42:03::
review.VERB d000.s000.t002 review%2:31:00::
objective.NOUN d000.s000.t003 objective%1:09:00::
```

Convert by yourself!

Training

- Edit train_one.bash
  - change memory size based on your machine

```
export LANG=en_US
java -mx1900m -cp $CP sg.edu.nus.comp.nlp.ims.implement.CTrainModel -prop $libdir/prop.xml -ptm $li
```

original

```
export LANG=en_US
java -Xmx30G -Xms30G -cp $CP sg.edu.nus.comp.nlp.ims.implement.CTrainModel -prop $libdir/prop.
```

in **coronation**

- Run train_one.bash

```
./train_one.bash train.xml train.key outputDir
```

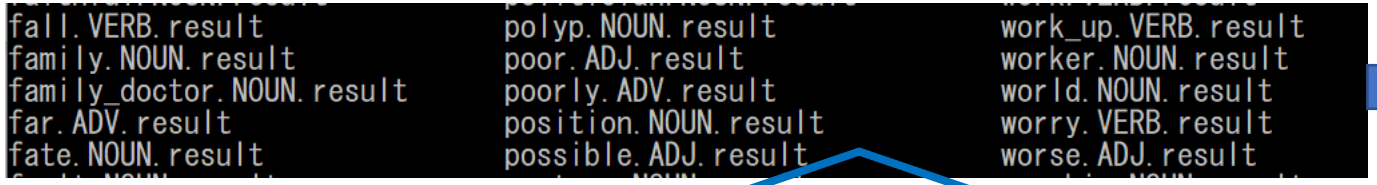
- Resulting models
  - *.stat.gz: statistic information
  - *.model.gz: model file

```
conduct.VERB.model.gz
conduct.VERB.stat.gz
conducting_wire.NOUN.stat.gz
conduction.NOUN.stat.gz
conductivity.NOUN.stat.gz
conductor.NOUN.model.gz
conductor.NOUN.stat.gz
cone.NOUN.stat.gz
```

Testing

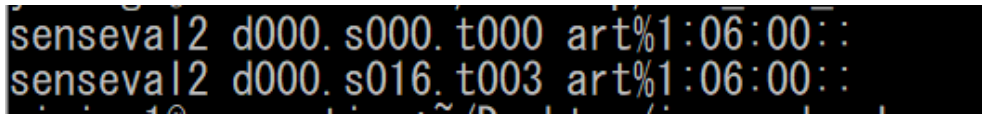
- Edit test_one.bash (same as train_one.bash)

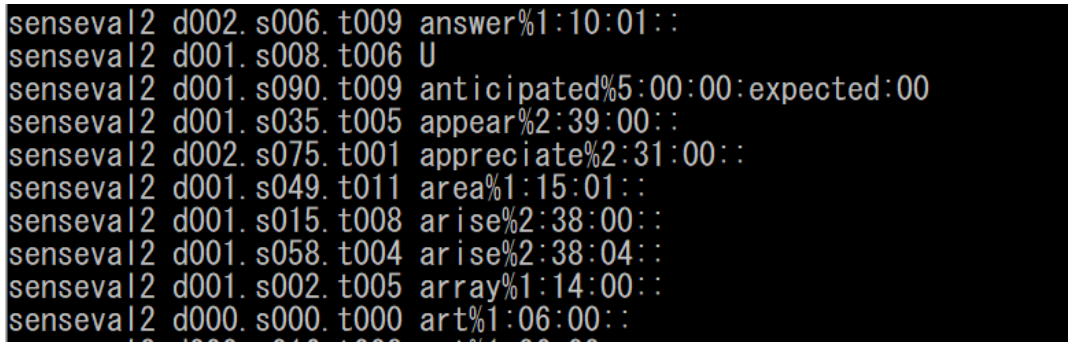
- Run test_one.bash to test fine-grained all-words WSD
  - testFine.bash doesn't work (don't produce any output)
  - test_one.bash → output different result files for each lexical item
    → concatenate those output files to create all-words output



concatenate

(e.g.) art.NOUN.result





Apply to Multilingual Setting 1

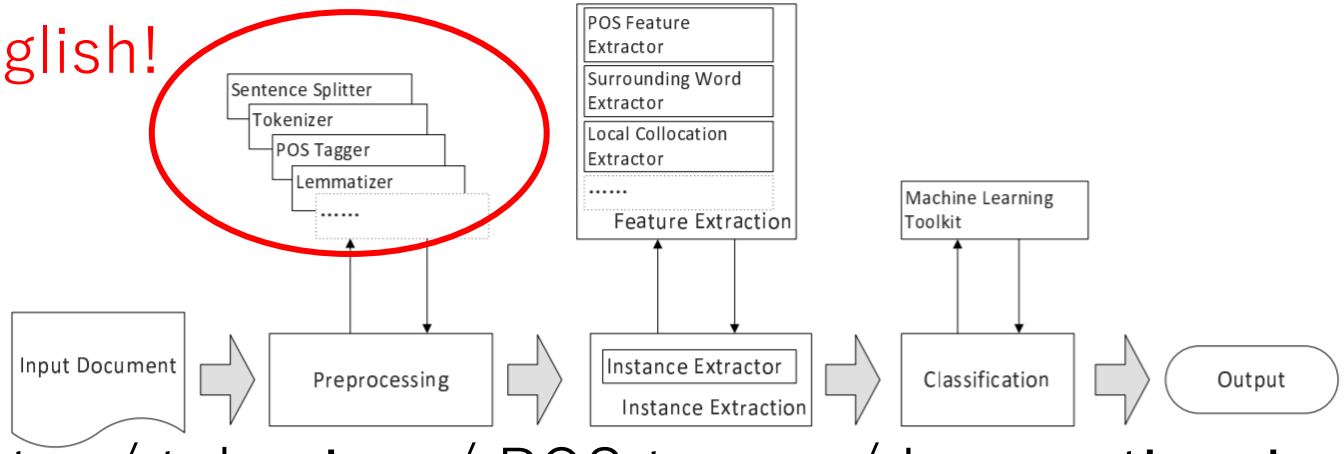
- Training Data → No manually sense-annotated data
  → Use automatically created data

  - Train-O-Matic (Pasini and Navigli, EMNLP2017):
    - ✓ sense tagged using BabelNet synset ids
    - ✓ available for English, Italian, Spanish, German, French, and Chinese
    - ✓ already converted to the proper xml format for IMS
  - OneSec (Scarlini et al., ACL2019):
    - ✓ sense tagged using BabelNet synset ids
    - ✓ available for English, Italian, Spanish, German, and French
    - ✓ already converted to the proper xml format for IMS

- Test Data → SemEval2013 task12, SemEval2015 task13

Apply to Multilingual Setting 2

Only work with English!



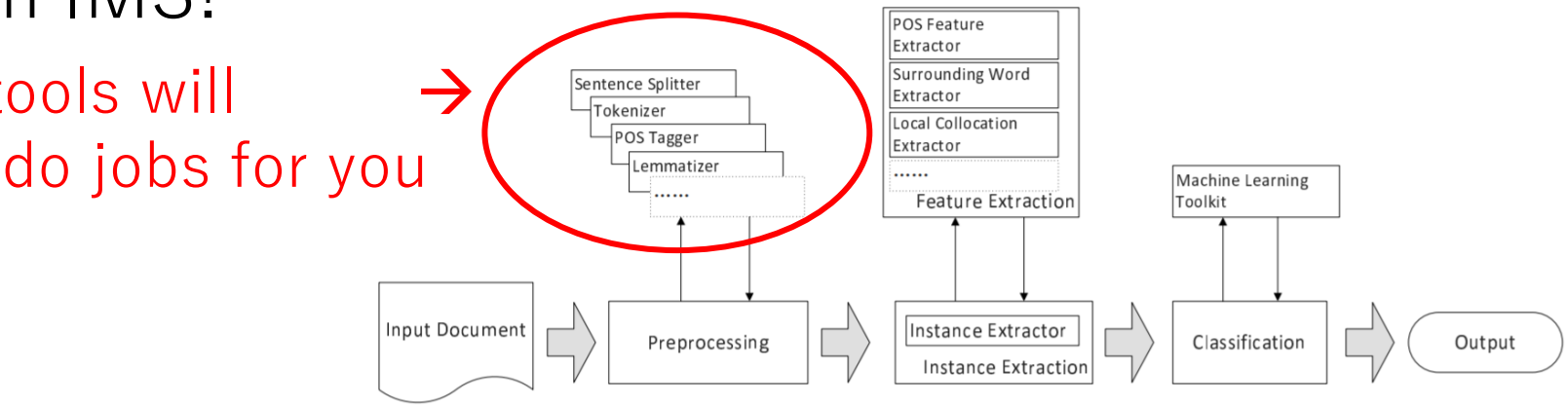
- Disable sentence splitter / tokenizer / POS tagger / Lemmatizer in IMS
  - edit (split, ssm, token, pos, ptm, lemma) flags in train_one.bash and test_one.bash
- Provide training / test data with POS and Lemma information (exist in the data)
  - (e.g.) Train-O-Matic Italian data

```
xml version="1.0" encoding="UTF-16"?>
orpus lang="english">
<lexelt item="antropofago.n"/>
<lexelt item="tozzo.n">
  <instance id="tozzo.4202043" score="57.99268286991197">
    <answer instance="tozzo.4202043" sensekey="bn:00010732n"/>
    <context>Esistono/esistere/v in/in/o commercio/commercio/n vari/vario/o tipi/tipo/n di/di/o zucchero/zucchero/n :/:/o Con/con/o questo,
    <head>tozzo/tozzo/n</head>
    di/di/o pane/pane/n e/e/o un/un/o bicchier/bicchier/n di/di/o vino/vino/n »/»/o ././o</context>
  </instance>
```

Apply to Plain Text

- English → just run IMS!

These tools will
do jobs for you →



- Other than English → lemmatize and POS-tag by yourself!
  - TreeTagger is a good option
    → support 25 languages including Italian, Spanish, French, etc.
    → can tokenize, lemmatize, POS-tag, etc.

Thank you for your attention.