AI Tutorial 2022
中文命名實體辨識

國立中央大學 電機工程學系
李龍豪 助理教授 (lhlee@ee.ncu.edu.tw)
林孜彌 研究生 (ltmdegf4@gmail.com)

https://github.com/NCUEE-NLPLab/AITutorial-2022-ChineseNER

Schedule

- Date: June 29th, 2022
- Schedule:

14:00	– 15:00	中文命名實體辨識研究發展
15:00	– 15:20	ROCLING-2022 Shared Task 介紹
15:20	– 15:40	break
15:40	– 16:40	BiLSTM-CRF 模型實作
16:40	– 17:00	Q & A 綜合討論

中文命名實體辨識研究發展

Named Entity Recognition (NER)

- Named Entity Recognition (NER) 命名實體辨識
  https://en.wikipedia.org/wiki/Named-entity_recognition

Named Entity Recognition (**NER**) (also known as (named) entity identification, entity chunking, and entity extraction) is a subtask of **information extraction** that seeks to **locate** and **classify** named entities mentioned in unstructured text into **pre-defined categories** such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.

CoNLL 2002/2003 Shared Task Language-Independent Named Entity Recognition

- **CoNLL** is a yearly conference organized by **SIGNLL (ACL's Special Interest Group on Natural Language Learning)**, focusing on theoretically, cognitively and scientifically motivated approaches to computational linguistics.
- CoNLL 2002/2003 分別舉辦語言獨立的命名實體辨識評測任務 (shared tasks)
- 兩年的任務共有4 種語言：2002 西班牙文和荷蘭文 / 2003 英文和德文
- 命名實體總共 4 類：persons (PER), locations (LOC), organizations (ORG) and miscellaneous names (MISC)

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Data Format from CoNLL 2002/2003 Tasks

- **BIO tagging schema**
  - B-XXX: the first word in a named entity of type XXX
  - I-XXX: all other words inside a named entity of type XXX
  - O: outside of named entity

[PER Wolff] , currently a journalist in [LOC Argentina] , played with [PER Del Bosque] in the final years of the seventies in [ORG Real Madrid] .

[ORG U.N.] official [PER Ekeus] heads for [LOC Baghdad] .

U.N.	NNP	I-NP	I-ORG
official	NN	I-NP	O
Ekeus	NNP	I-NP	I-PER
heads	VBZ	I-VP	O
for	IN	I-PP	O
Baghdad	NNP	I-NP	I-LOC
.	.	O	O

Wolff	B-PER
,	O
currently	O
a	O
journalist	O
in	O
Argentina	B-LOC
,	O
played	O
with	O
Del	B-PER
Bosque	I-PER

中文命名實體辨識 (Chinese NER)

- Chinese NER is relatively difficult
  - Chinese language is logographic
  - No conventional features (e.g. capitalization)
  - A lack of delimiters between characters
  - Incorrectly segmented entity boundaries will cause error propagation

土地公有政策
→ 土地 (land) + 公有 (public) + 政策 (policy)
→ ？土地公有 + 政策
→ ？？ 土地公 + 有 + 政策

思覺失調症 (schizophrenia)
→ 思覺 (thinking and feeling)
+ 失調 (disorder)
+ 症 (disease)

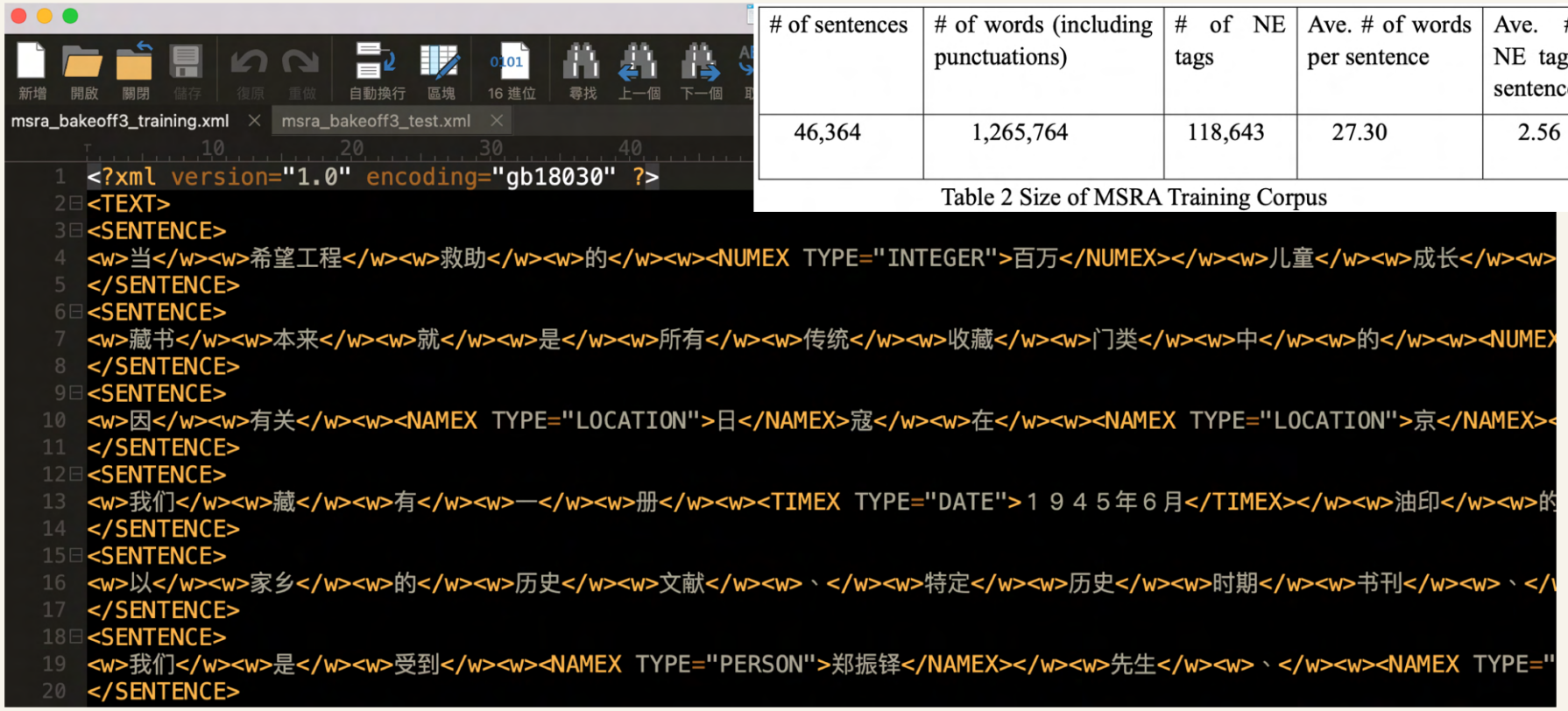
SIGHAN 2006 Bakeoff – Chinese NER

- **SIGHAN** (ACL's **Special Interest Group on Chinese Language Processing**)
- Organizer: Gina-Anne Levow (Dept. of Linguistics, Univ. of Washington)
- 兩個任務：斷詞任務 (word segmentation) & 命名實體任務 (named entity recognition)
- 命名實體共有 3 類： 人名 (PER)、地名(LOC)、組織名 (ORG)
- 資料來源 (簡體/繁體)：

Source	Encodings	Training (Wds/Types)	Test (Wds/Types)
CITYU	BIG5HKSCS/Unicode	1.6M/76K	220K/23K
CKIP	BIG5/Unicode	5.5M/146K	91K/15K
LDC	Unicode	632K (est. wds)	61K (est. wds)
MSRA	GB18030/Unicode	1.3M/63K	100K/13K
UPUC	GB/Unicode	509K/37K	155K/17K

Gina-Anne Levow. 2006. The Third International Chinese Language Processing Bakeoff: Word Segmentation and Named Entity Recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117.

MSRA NE Corpus from SIGHAN 2006 Bakeoff



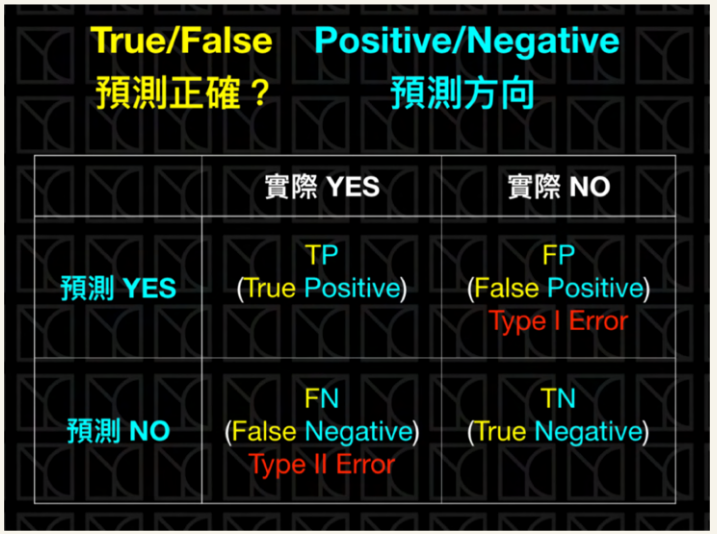
# of sentences	# of words (including punctuations)	# of NE tags	Ave. # of words per sentence	Ave. # of NE tags per sentence
46,364	1,265,764	118,643	27.30	2.56

Table 2 Size of MSRA Training Corpus

NER Model Evaluation (模型評測)

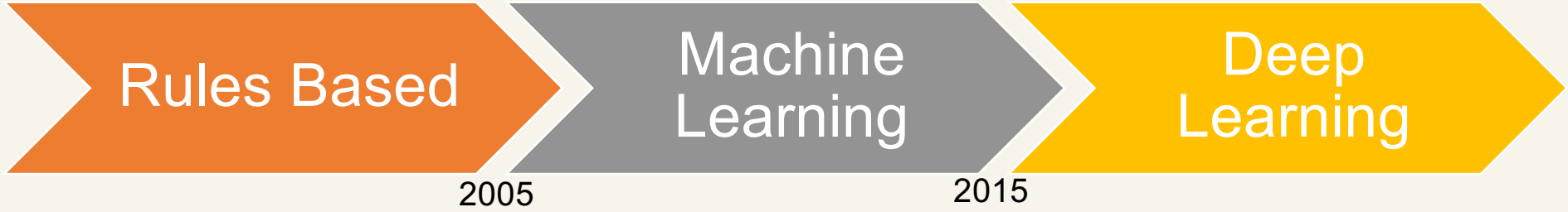
- **Metrics: Precision / Recall / F1 Score**
  - Precision (Pre.) = TP / (TP+FP)
  - Recall (Rec.) = TP / (TP + FN)
  - F1 Score = 2 * Pre * Rec / (Pre + Rec)

Confusion Matrix (混淆矩陣)



(資料來源: https://www.ycc.idv.tw/confusion-matrix.html)

NER Approaches

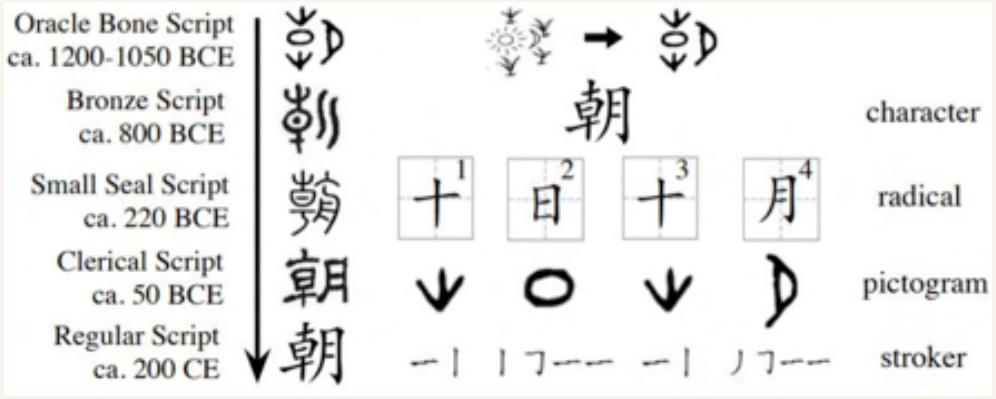


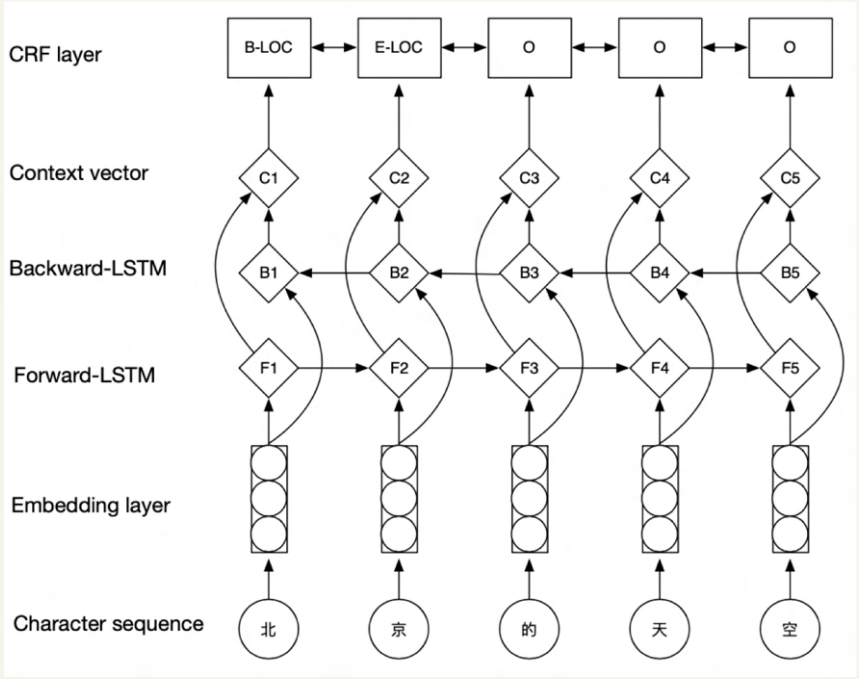
- Heuristic Rules
- Lexicons

- Hidden Markov Model (HMM)
- Maximum Entropy Markov Model (MEMM)
- Conditional Radom Field (CRF)

- BiLSTM-CRF with radical features
- Lattice LSTM
- ME-CNER
- Gazetteers
- BERT
- ME-MGNN

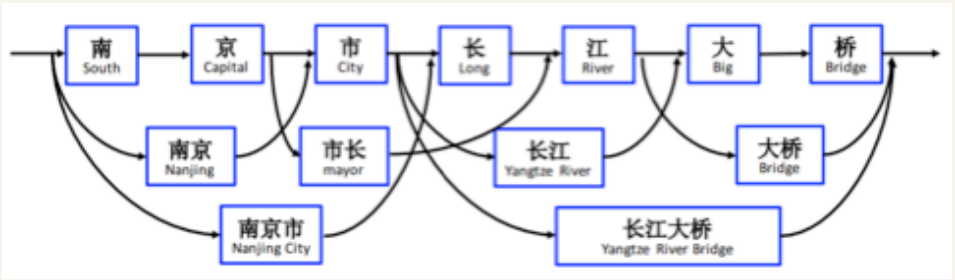
BiLSTM-CRF with Radical Features (Dong et al., 2016)

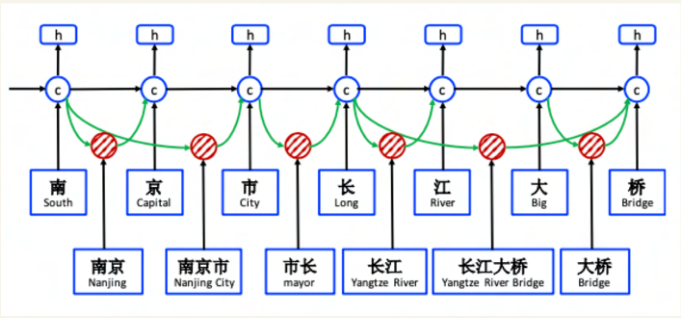


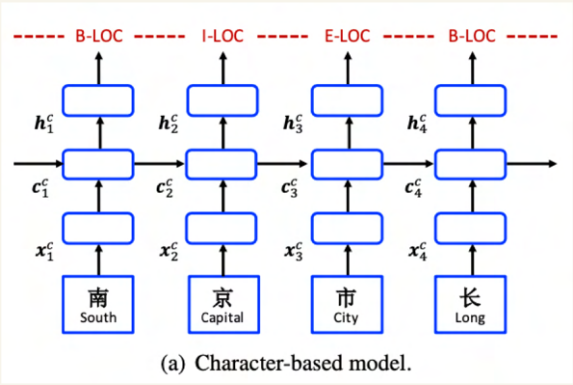


Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based LSTM-CRF with Radical-level features for Chinese Named Entity Recognition. In Proceedings of the 24th International Conference of Computer Processing of Oriental Languages, pages 239-250.

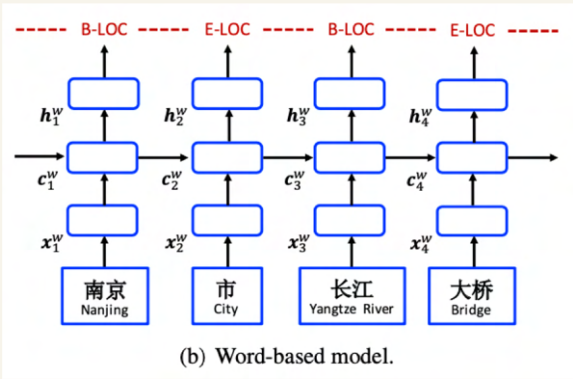
Lattice LSTM (Zhang and Yang, 2018)



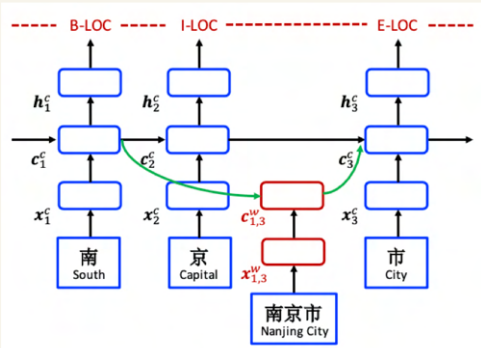




(a) Character-based model.



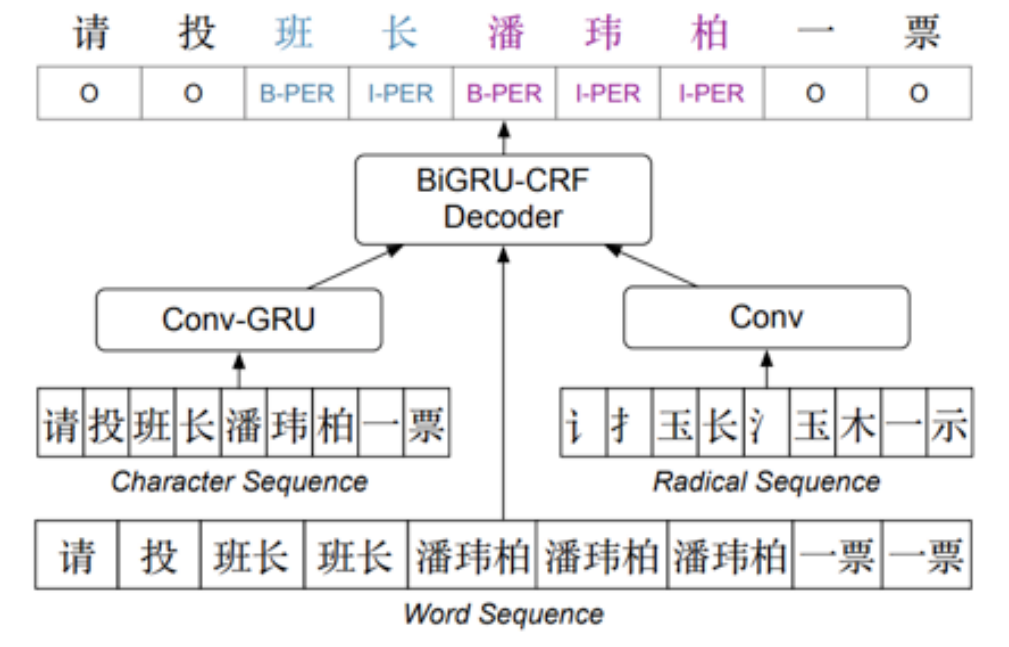
(b) Word-based model.

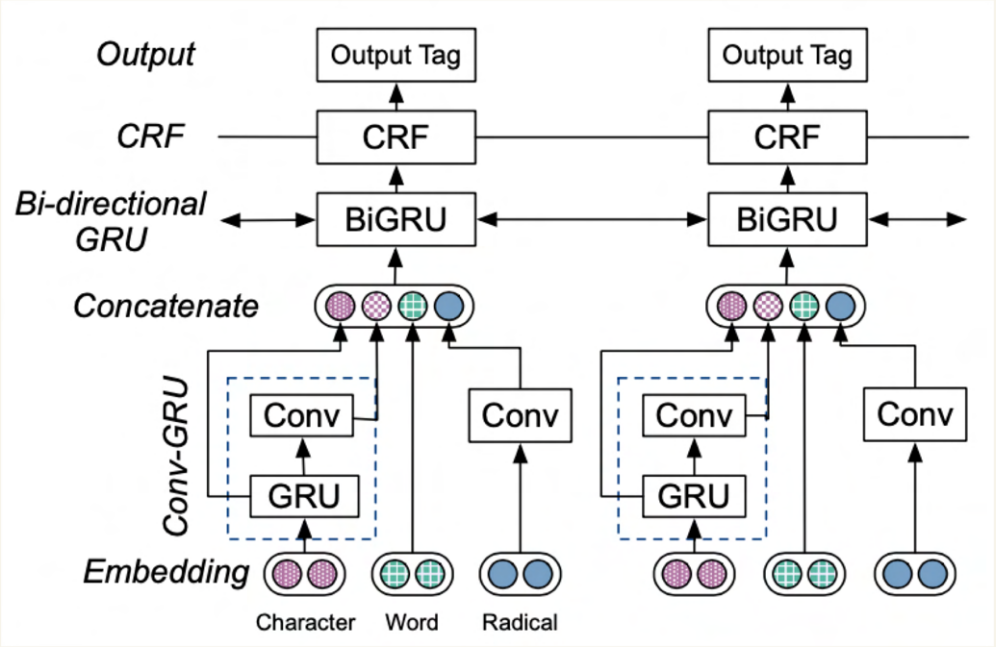


(c) Lattice model.

Yue Zhang and Jie Yang. 2018. Chinese NER Using Lattice LSTM. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1554–1564,.

ME-CNER (Xu et al., 2019)

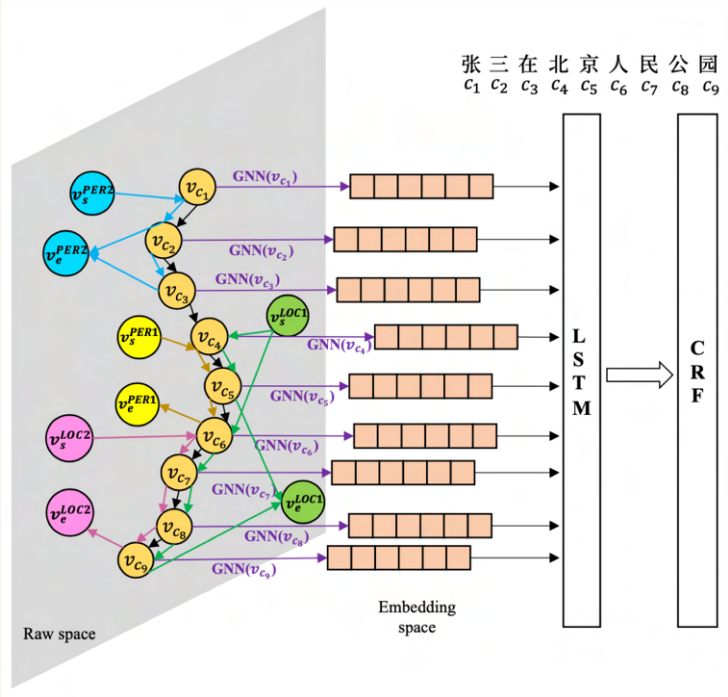




Canwen Xu, Feiyang Wang, Jialong Han, and Chenliang Li. 2019. Exploiting Multiple Embeddings for Chinese Named Entity Recognition. In *Proceedings of the 28th ACM Conference on Information and Knowledge Management*, pages 2269-2272.

Gazetteers (Ding et al., 2019)

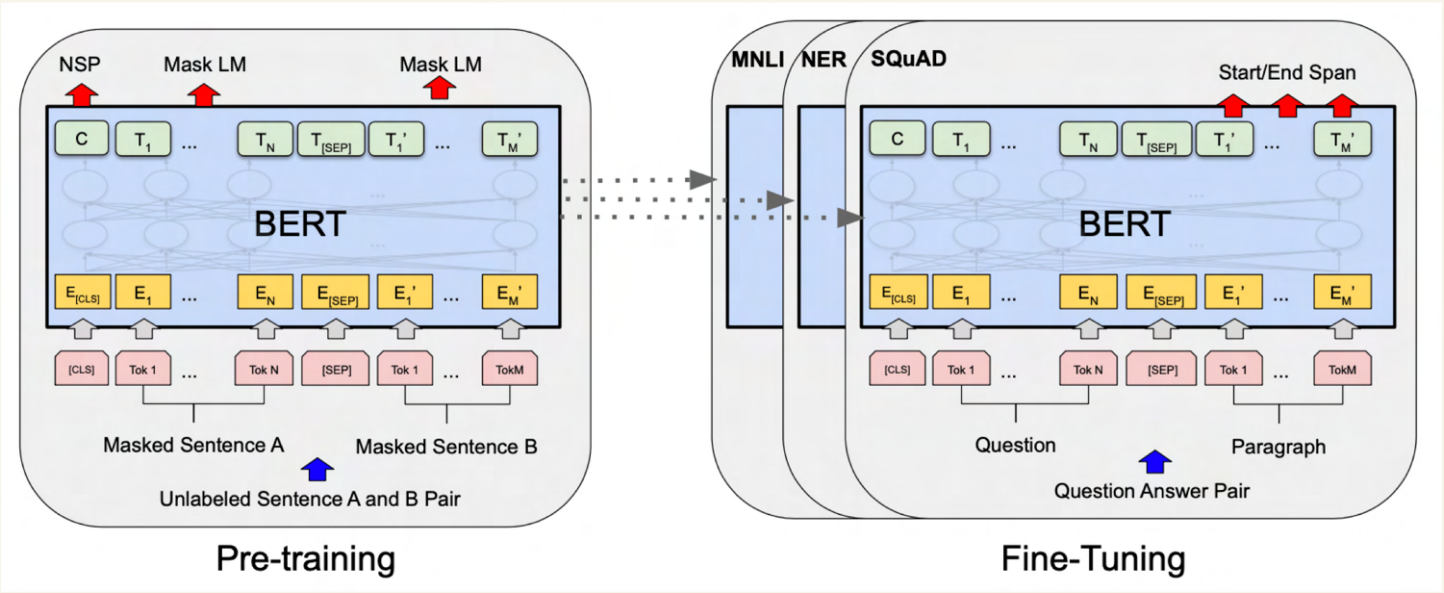
- Neural Multi-graph model

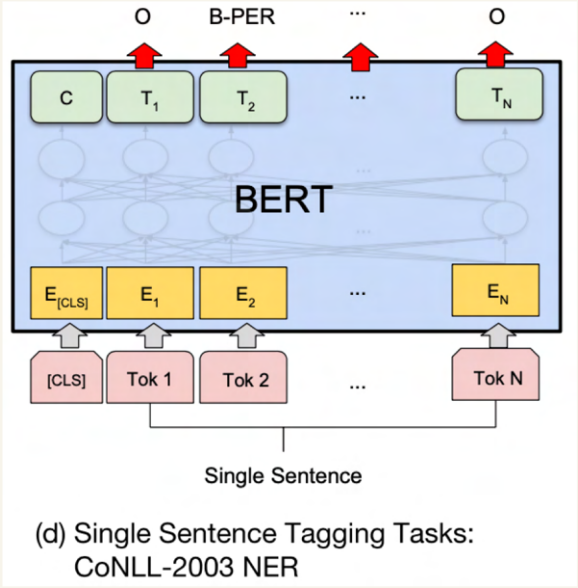


Ruixue Ding, Pengjun Xie, Xiaoyan Zhang, Wei Lu, Linlin Li, and Luo Si. 2019. A Neural Multi-digraph Model for Chinese NER with Gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1462–1467.

BERT (Devlin et al., 2019)

- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (**BERT**)

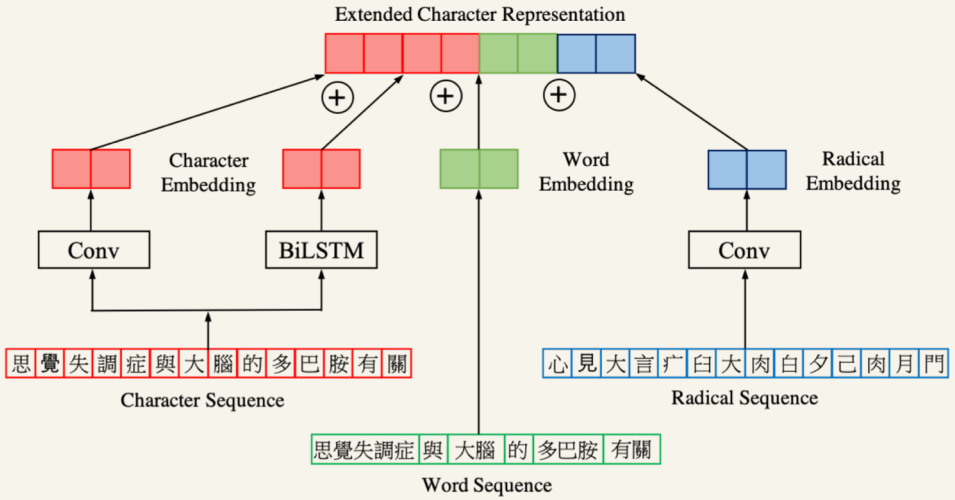


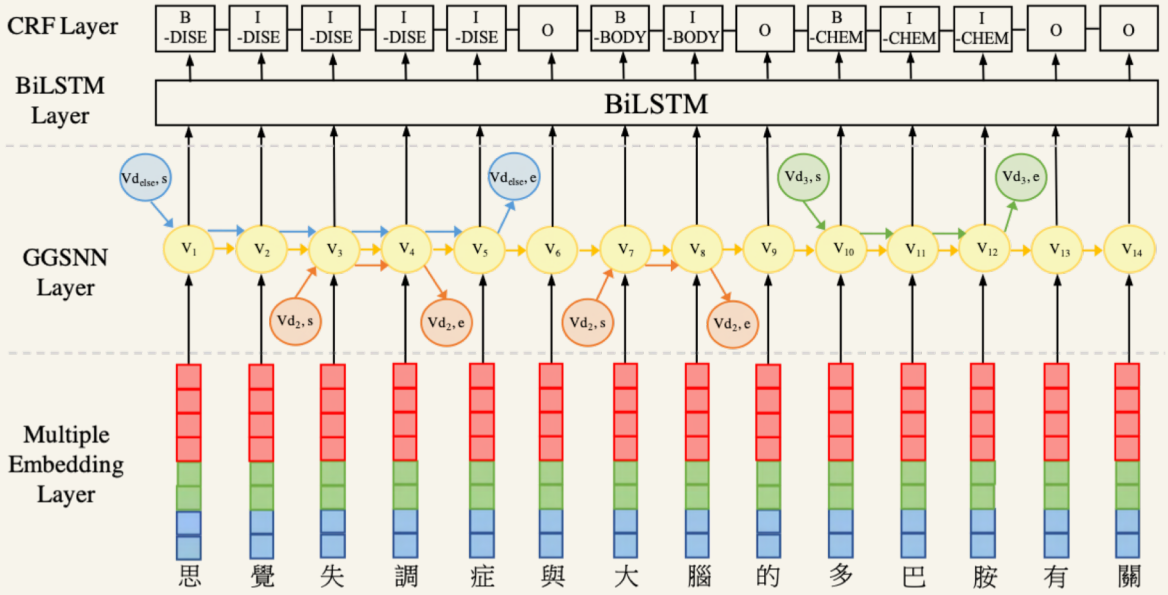


Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

ME-MGNN (Lee and Lu, 2021)

- Multiple Embeddings enhanced Multi-Graph Neural Networks (ME-MGNN)





Lung-Hao Lee, and Yi Lu. 2021. Multiple Embeddings Enhanced Multi-Graph Neural Networks for Chinese Healthcare Named Entity Recognition. *IEEE Journal of Biomedical and Health Informatics*, 25(7): 2801-2810.

Model Performance Comparisons

- Experimental results on Chinese HealthNER Corpus (Lee and Lu, 2021)

Model	Precision	Recall	F1
BiLSTM-CRF with radical (Dong et al., 2016)	0.7038	0.7277	0.7156
BERT (Devlin et al., 2019)	0.7145	**0.7636**	0.7382
ME-CNER (Xu et al., 2019)	0.7368	0.7462	0.7415
Gazetteers (Ding et al., 2019)	0.7300	0.7556	0.7426
Lattice (Ding et al., 2018)	0.7469	0.7576	0.7522
ME-MGNN (Lee and Lu, 2021)	**0.7546**	0.7576	**0.7569**

- On a Nvidia DGX-1 sever using a V100 GPU with 512G Memory
  - The Lattice model spent about 6 days and 4 hours for training the NER model
  - ME-MGNN model is about 23 hours

Model Summary: BiLSTM-CRF

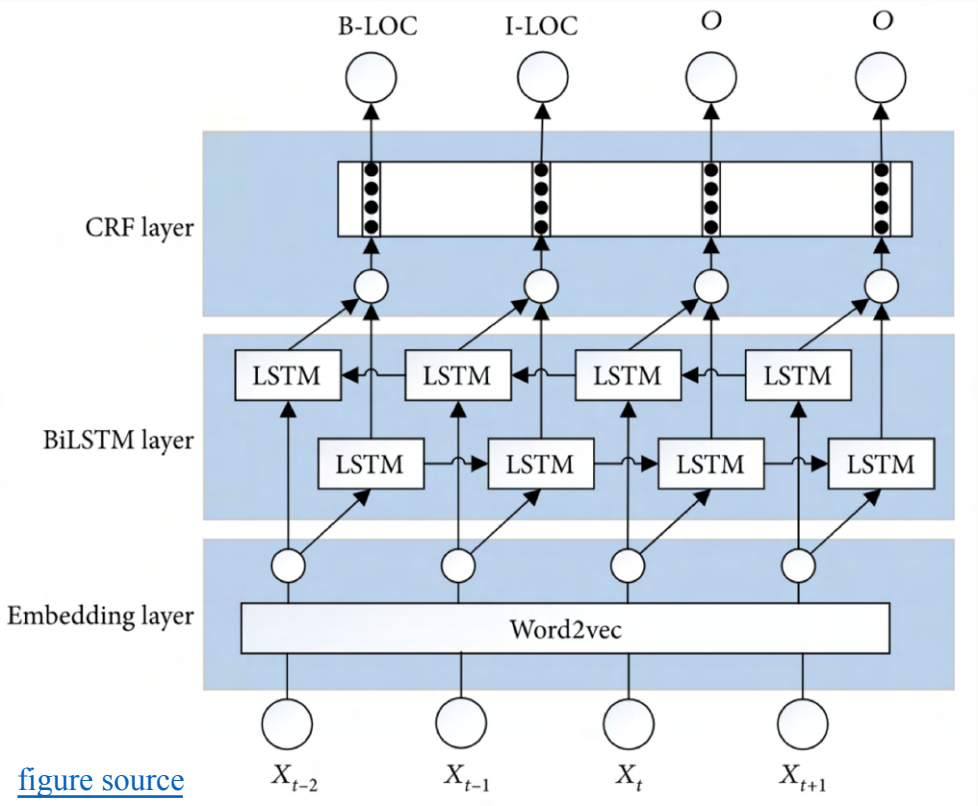


figure source

SemEval-2022 Task 11: MultiCoNER
Track 9 – Chinese (ZH) Results

	Team Name	Affiliation	Username	Submission ID	F1
1	USTC-NELSLIP	NELSLIP, University of Science and Technology of China	bdchen	1086812	0.8169
2	CASIA	Institute of automation, Chinese Academy of Sciences	Fujia	1087074	0.797
3	OPDAI	Interactive Entertainment Group of Netease Inc.	zackchen	1086527	0.7954
4	DAMO-NLP	Alibaba Group, ShanghaiTech University, Zhejiang University, SUTD	oriuta	1086413	0.7806
5	NetEase.AI	NetEase	Mrhou	1086029	0.7777
6	CMB AI Lab	CMB AI Lab	CestDoDo	1084842	0.7636
7	NCUEE-NLP	National Central University	smallbowl	1086596	0.7418
8	QTrade AI	QTrade(深圳苹果树数据科技有限公司)	omnifish_ygb	1087001	0.74
9	CSECU-DSG	University of Chittagong	csecudsg	1086326	0.6722
10	Multilinguals	IIIT-HYDERABAD	AmitPandey	1081673	0.6695
11	L3i	University of La Rochelle	emanuela.boros	1081273	0.6691
12	Sliced	IT University of Copenhagen	sliced	1086838	0.6521
13	Infrrd.ai	infrrd.ai	au1206	1086799	0.6468
14	MaChAmp	IT University of Copenhagen	robvanderg	1086151	0.6381
15	EURECOM	EURECOM	jplu	1091476	0.634
16	RACAI	Research Institute for Artificial Intelligence "Mihai Draganescu", Romanian	pvf	1080171	0.627

https://multiconer.github.io/results

Related Discussions

- Different Granularities
  → Character-based or Word-based

- Different Representations
  → Word2vec, GloVe, fastText, ... etc.

- Different Neural Architecture
  → Stacked-based, Graph-based or Transformer-based

ROCLING-2022 Shared Task 介紹

https://rocling2022.github.io/#sharedTask

Organizers: 中央電機 李龍豪、中央電機 陳昭沂、元智資管 禹良治、師大圖資 曾元顯

Chinese Healthcare NER

- In the digital era, healthcare information-seeking users usually search had browse web content in click-through trails to obtain healthcare-related information before making a doctor's appointment for diagnosis and treatment.

- For example,
  - 三酸甘油酯 (triglyceride) => 化學物質
  - 電腦斷層掃描 (computed tomography; CT) => 檢查
  - 靜脈免疫球蛋白注射 (intravenous immunoglobulin; IVIG) => 治療

Named Entity Types

類型	描述	範例
人體 Body (BODY)	The whole physical structure that forms a person or animal including biological cells, organizations, organs and systems.	“細胞核” (nucleus), “神經組織” (nerve tissue), “左心房” (left atrium), “脊髓” (spinal cord), “呼吸系統” (respiratory system)
症狀 Symptom (SYMP)	Any feeling of illness or physical or mental change that is caused by a particular disease.	“流鼻水” (rhinorrhea), “咳嗽” (cough), “貧血” (anemia), “失眠” (insomnia), “心悸” (palpitation), “耳鳴” (tinnitus)
器材 Instrument (INST)	A tool or other device used for performing a particular medical task such as diagnosis and treatments.	“血壓計” (blood pressure meter), “達文西手臂” (DaVinci Robots), “體脂肪計” (body fat monitor), “雷射手術刀” (laser scalpel)
檢驗 Examination (EXAM)	The act of looking at or checking something carefully in order to discover possible diseases.	“聽力檢查” (hearing test), “腦電波圖” (electroencephalography; EEG), “核磁共振造影” (magnetic resonance imaging; MRI)
化學物質 Chemical (CHEM)	Any basic chemical element typically found in the human body.	“去氧核糖核酸” (deoxyribonucleic acid; DNA), “糖化血色素” (glycated hemoglobin), “膽固醇” (cholesterol), “尿酸” (uric acid)
疾病 Disease (DISE)	An illness of people or animals caused by infection or a failure of health rather than by an accident.	“小兒麻痺症” (poliomyelitis; polio), “帕金森氏症” (Parkinson’s disease), “青光眼” (glaucoma), “肺結核” (tuberculosis)
藥物 Drug (DRUG)	Any natural or artificially made chemical used as a medicine	“阿斯匹靈” (aspirin), “普拿疼” (acetaminophen), “青黴素” (penicillin), “流感疫苗” (influenza vaccination)
營養品 Supplement (SUPP)	Something added to something else to improve human health.	“維他命” (vitamin), “膠原蛋白” (collagen), “益生菌” (probiotics), “葡萄糖胺” (glucosamine), “葉黃素” (lutein)
治療 Treatment (TREAT)	A method of behavior used to treat diseases	“藥物治療” (pharmacotherapy), “胃切除術” (gastrectomy), “標靶治療” (targeted therapy), “外科手術” (surgery)
時期 Time (TIME)	Element of existence measured in minutes, days, years	“嬰兒期” (infancy), “幼兒時期” (early childhood), “青春期” (adolescence), “生理期” (on one’s period), “孕期” (pregnancy)

Shared Task Examples

- Example 1
  - Input: 修復肌肉與骨骼最重要的便是熱量、蛋白質與鈣質。
  - Output: O, O, B-BODY, I-BODY, O, B-BODY, I-BODY, O, O, O, O, O, O, O, O, O, O, B-CHEM, I-CHEM, I-CHEM, O, B-CHEM, I-CHEM, O
- Example 2
  - Input: 如何治療胃食道逆流症？
  - Output: O, O, O, O, B-DISE, I-DISE, I-DISE, I-DISE, I-DISE, I-DISE, O

Chinese HealthNER Corpus

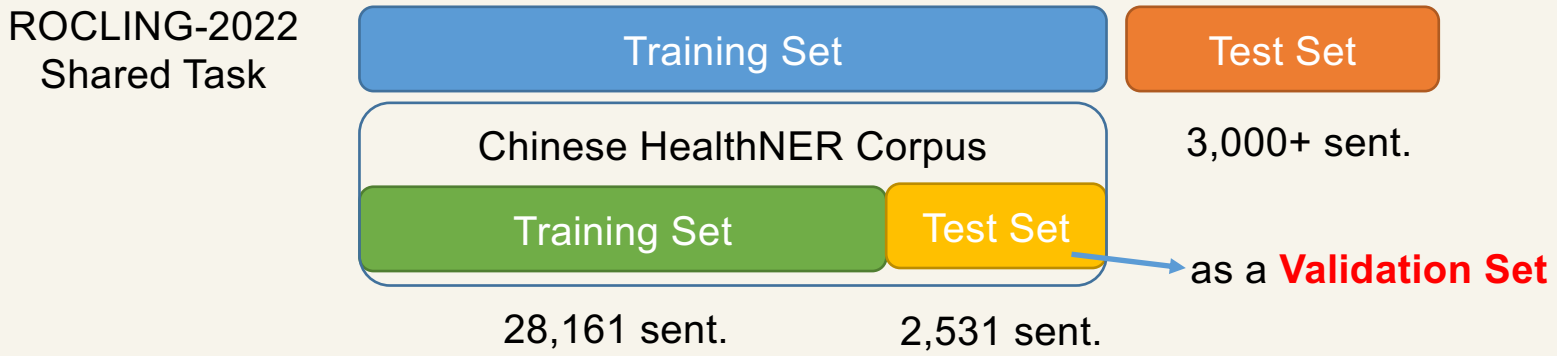
https://github.com/NCUEE-NLPLab/Chinese-HealthNER-Corpus

- Three undergraduate students majoring in Chinese language were trained in **word segmentation** and the **named entity tagging** task. Inter-annotator agreement is 84.1%. All annotators were asked to discuss differences and seek consensus.
- In summary, our constructed corpus includes 30,692 sentences with a total of around 1.5 million characters or 91.7 thousand words.


```
{
        "id": "00002",
        "genre": "sm",
        "sentence": "如何治療胃食道逆流症？",
        "word": ["如何", "治療", "胃食道逆流症", "？"],
        "word_label": ["O", "O", "DISE", "O"],
        "character": ["如", "何", "治", "療", "胃", "食", "道", "逆", "流", "症", "？"],
        "character_label": ["O", "O", "O", "O", "B-DISE", "I-DISE", "I-DISE", "I-DISE", "I-DISE", "I-DISE", "O"]
}
```


Lung-Hao Lee, and Yi Lu. 2021. Multiple Embeddings Enhanced Multi-Graph Neural Networks for Chinese Healthcare Named Entity Recognition. *IEEE Journal of Biomedical and Health Informatics*, 25(7): 2801-2810.

Training and Test Set



- Training Set: **Chinese HealthNER Corpus** (Lee and Lu, 2021)
  It includes 30,692 sentences with a total around 1.5 million characters or 91.7 thousand words. After manual annotation, we have 68,460 named entities across 10 entity types: body, symptom, instrument, examination, chemical, disease, drug, supplement, treatment, and time.
- Test Set: at least 3,000 Chinese sentences.

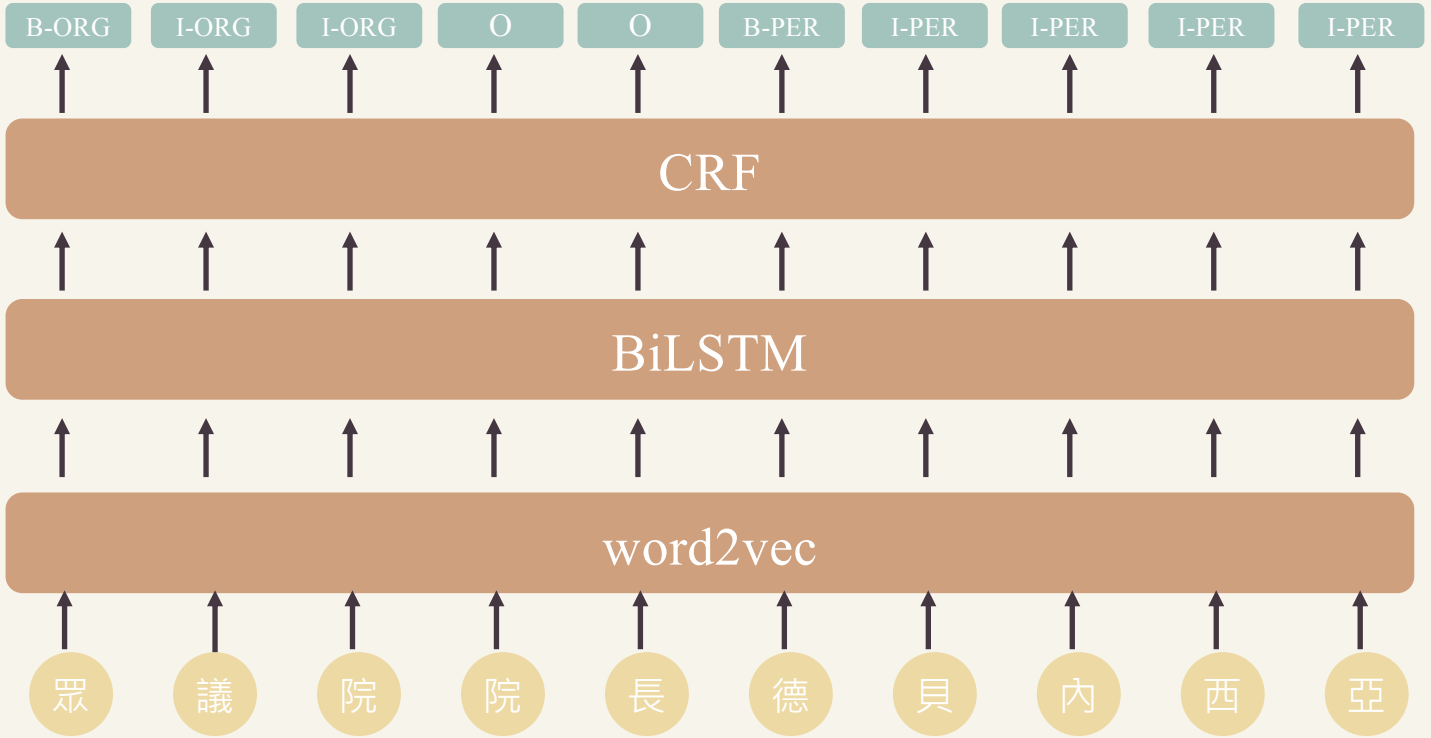
Call for Participation

- Registration: https://reurl.cc/0p48DY (Due: August 20, 2022)

- Important Dates
  - ~~Release of training data: April 15, 2022~~
  - Release of test data: August 31, 2022
  - Testing results submission due: September 2, 2022
  - Release of evaluation results: September 5, 2022
  - System description paper due: September 20, 2022
  - Notification of Acceptance: September 30, 2022
  - Camera-ready deadline: October 7, 2022

BiLSTM-CRF 模型實作

BiLSTM+CRF 架構介紹

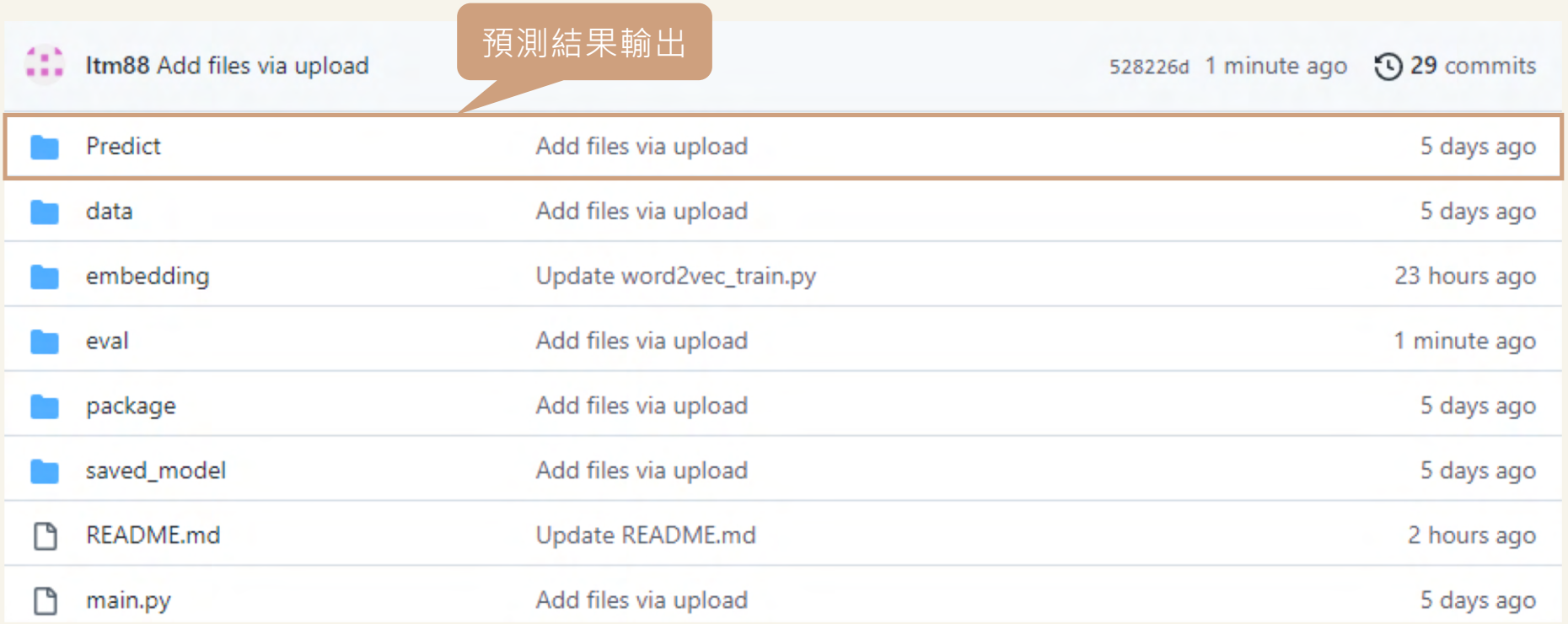


目錄

- Steps 1. GitHub 資料夾介紹
- Steps 2. 虛擬環境建置
- Steps 3. 詞嵌入訓練
- Steps 4. BiLSTM+CRF 訓練
- Steps 5. BiLSTM+CRF 測試
- Steps 6. 模型效能評估

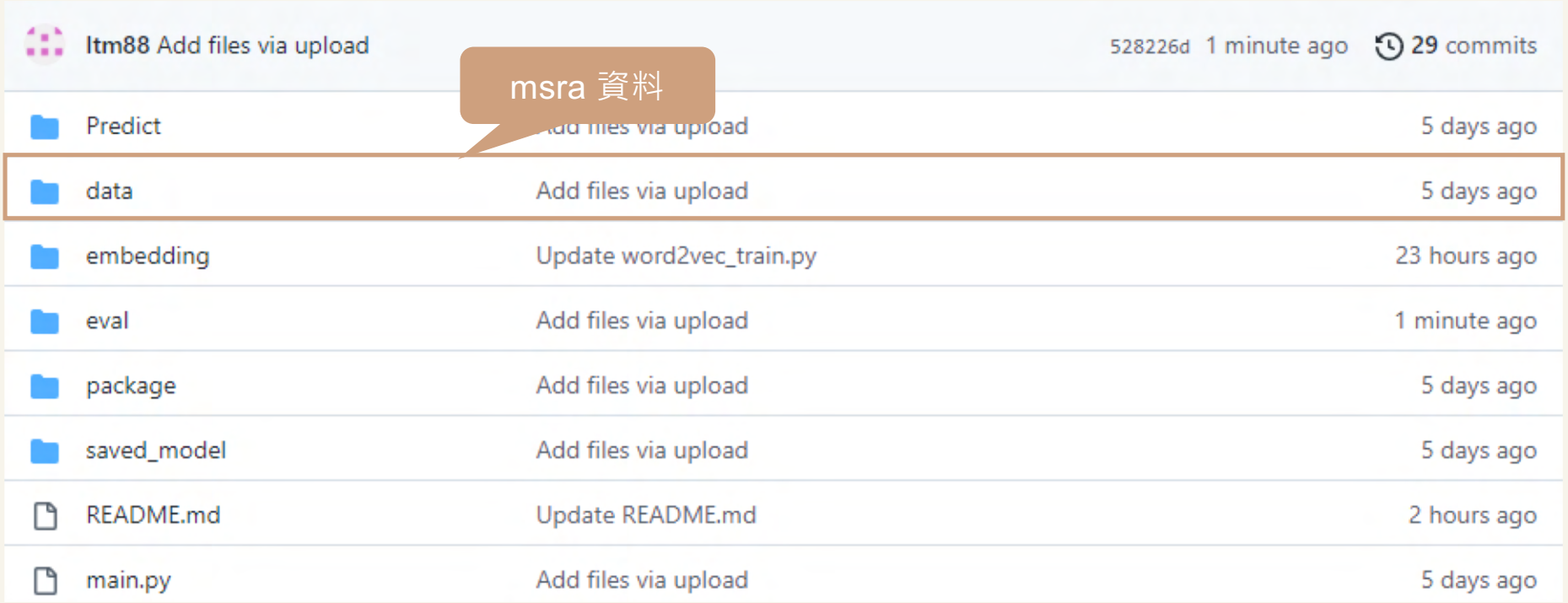
Steps 1. GitHub 資料夾介紹

GitHub 資料夾介紹



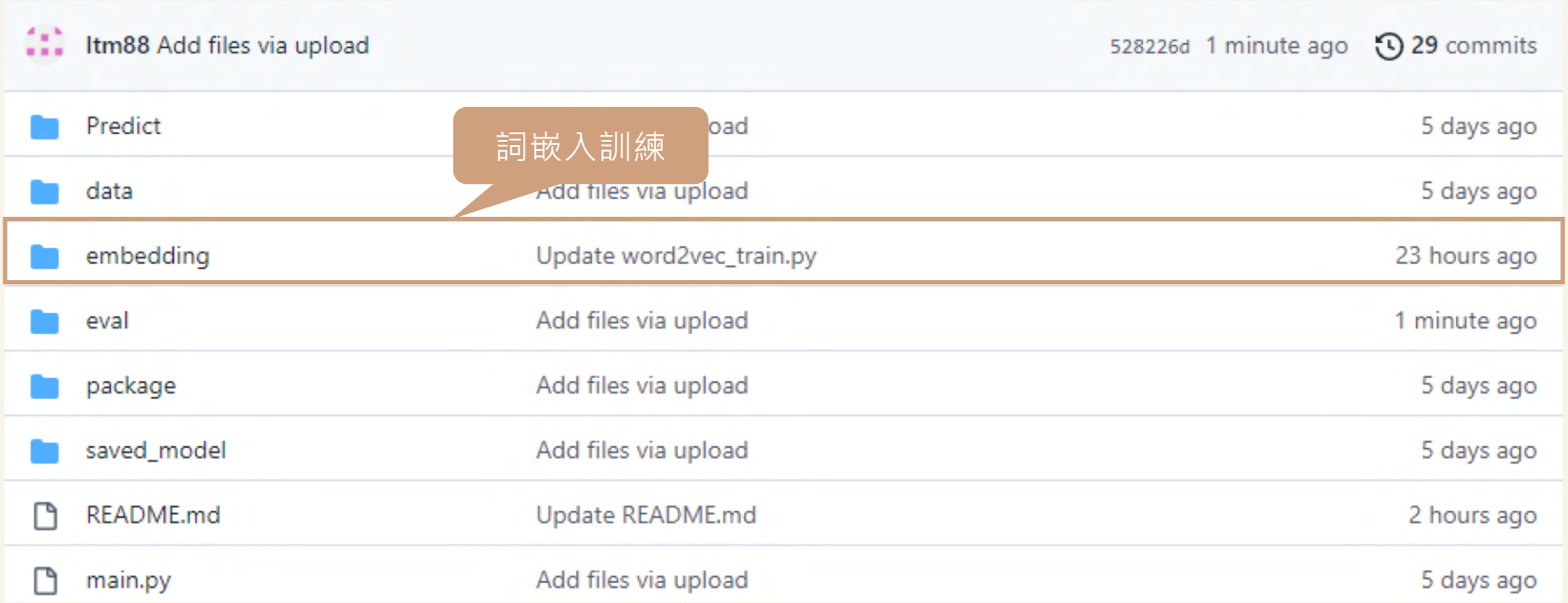
https://github.com/NCUEE-NLPLab/AITutorial-2022-ChineseNER

GitHub 資料夾介紹



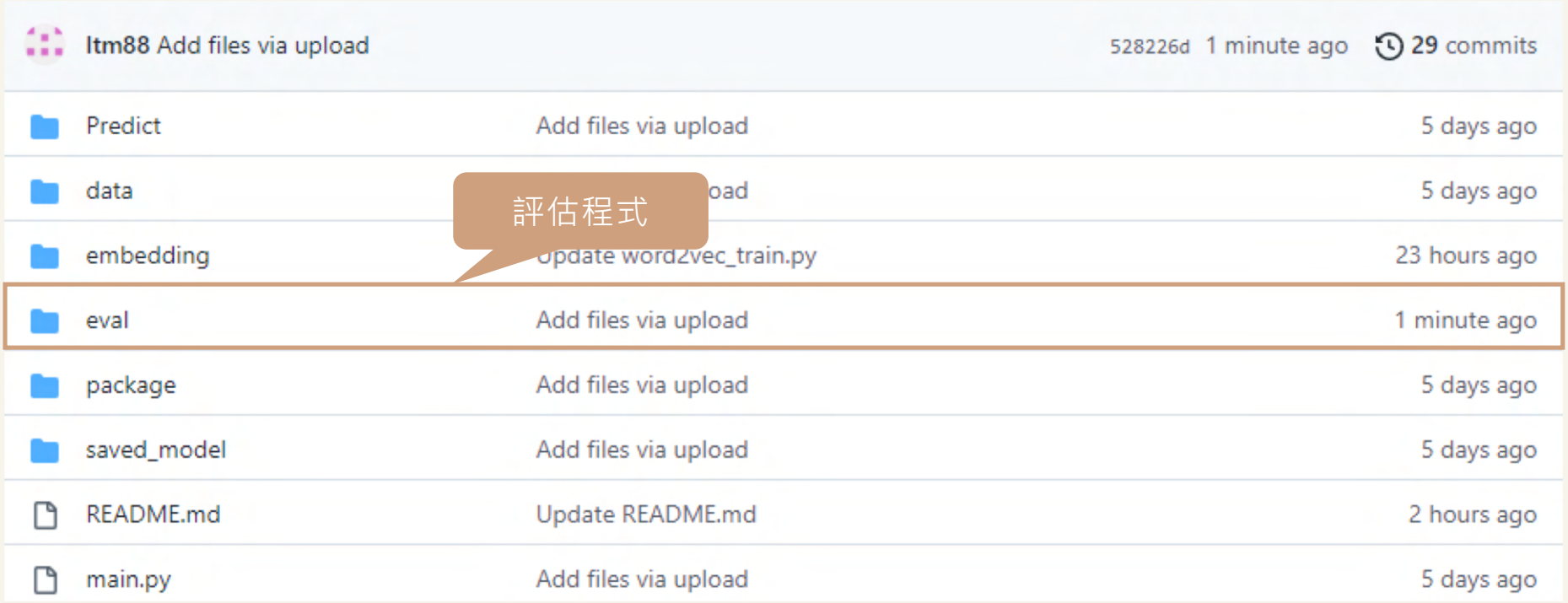
https://github.com/NCUEE-NLPLab/AITutorial-2022-ChineseNER

GitHub 資料夾介紹



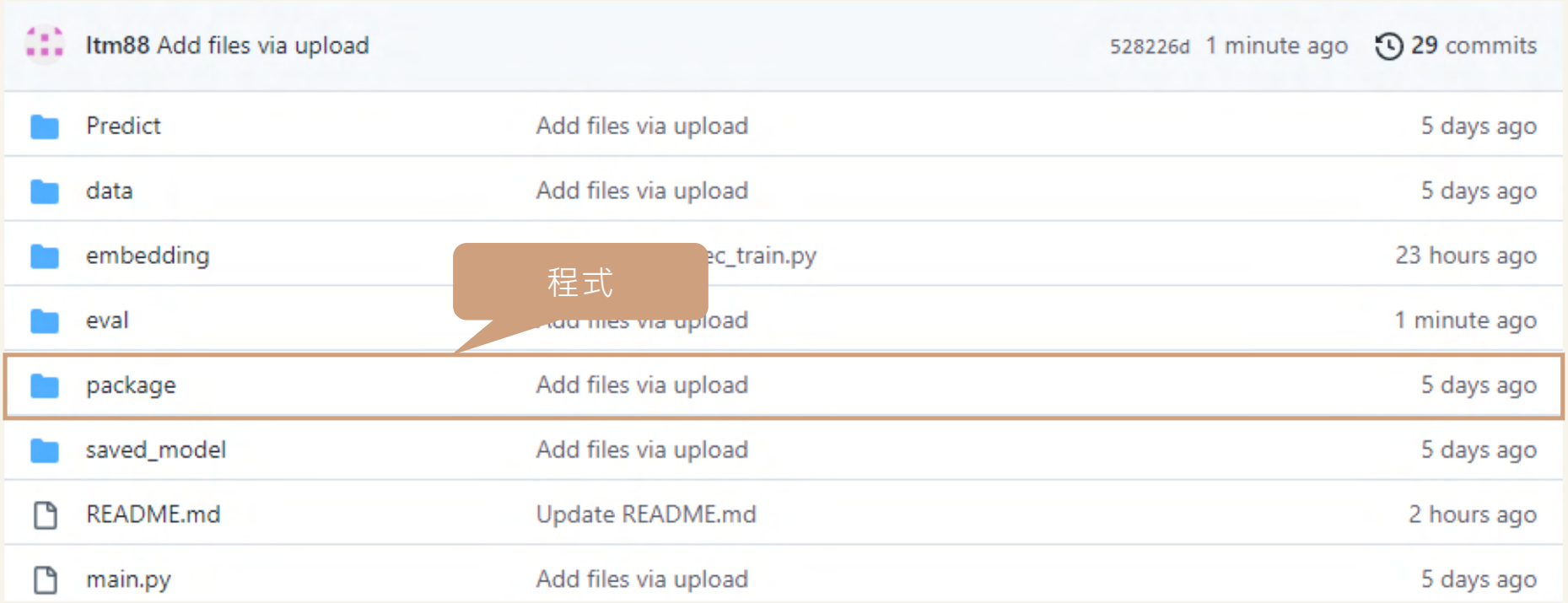
https://github.com/NCUEE-NLPLab/AITutorial-2022-ChineseNER

GitHub 資料夾介紹



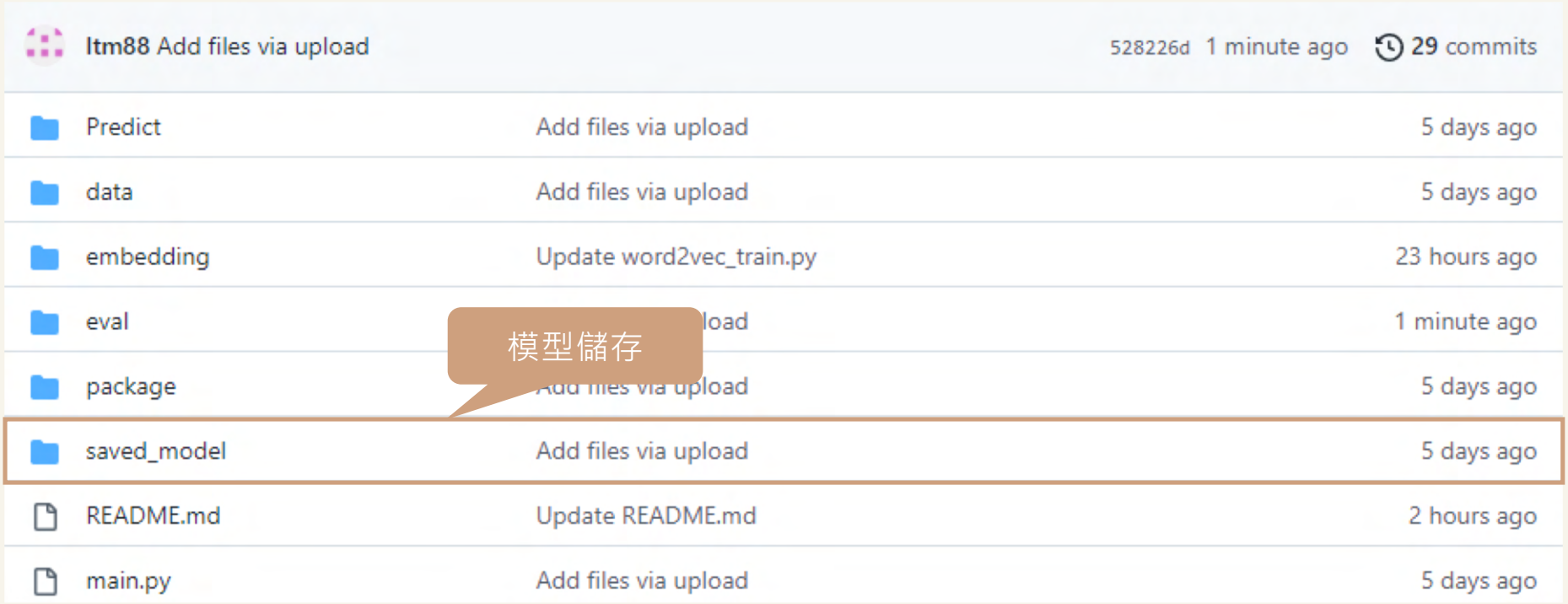
https://github.com/NCUEE-NLPLab/AITutorial-2022-ChineseNER

GitHub 資料夾介紹



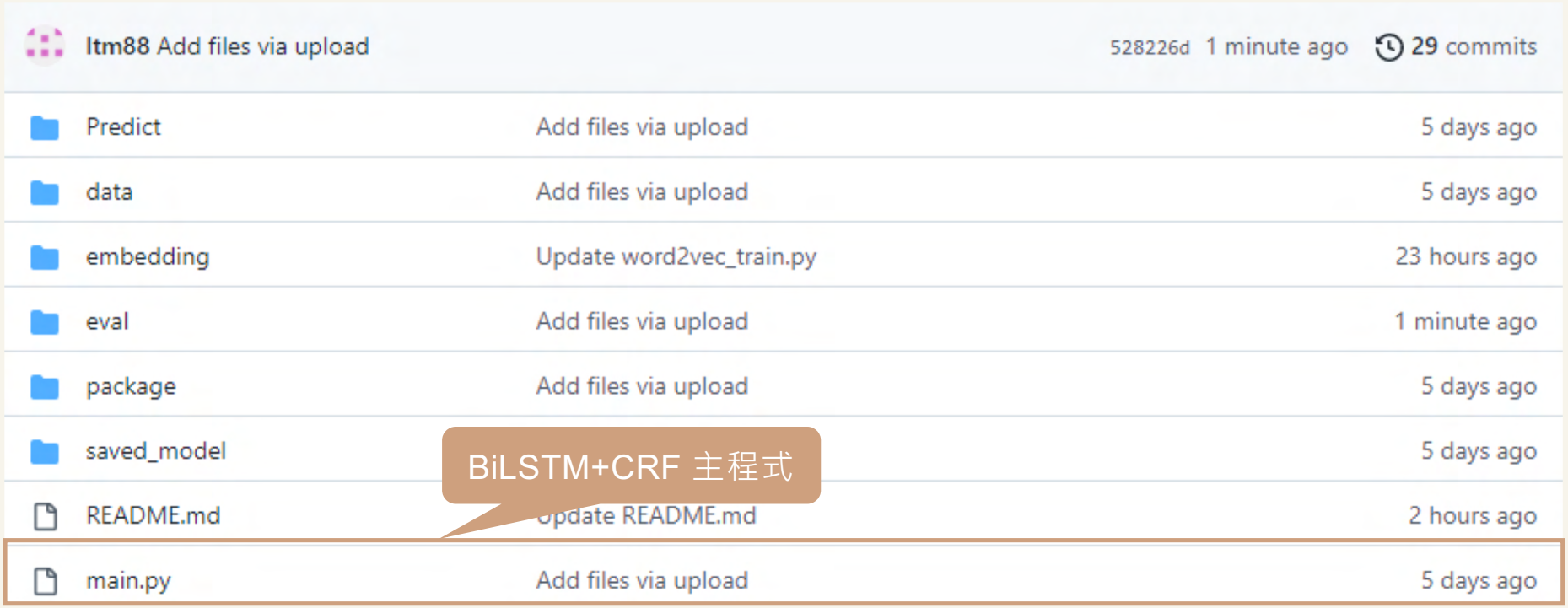
https://github.com/NCUEE-NLPLab/AITutorial-2022-ChineseNER

GitHub 資料夾介紹



https://github.com/NCUEE-NLPLab/AITutorial-2022-ChineseNER

GitHub 資料夾介紹



https://github.com/NCUEE-NLPLab/AITutorial-2022-ChineseNER

Steps 2. 虛擬環境建置

虛擬環境建置

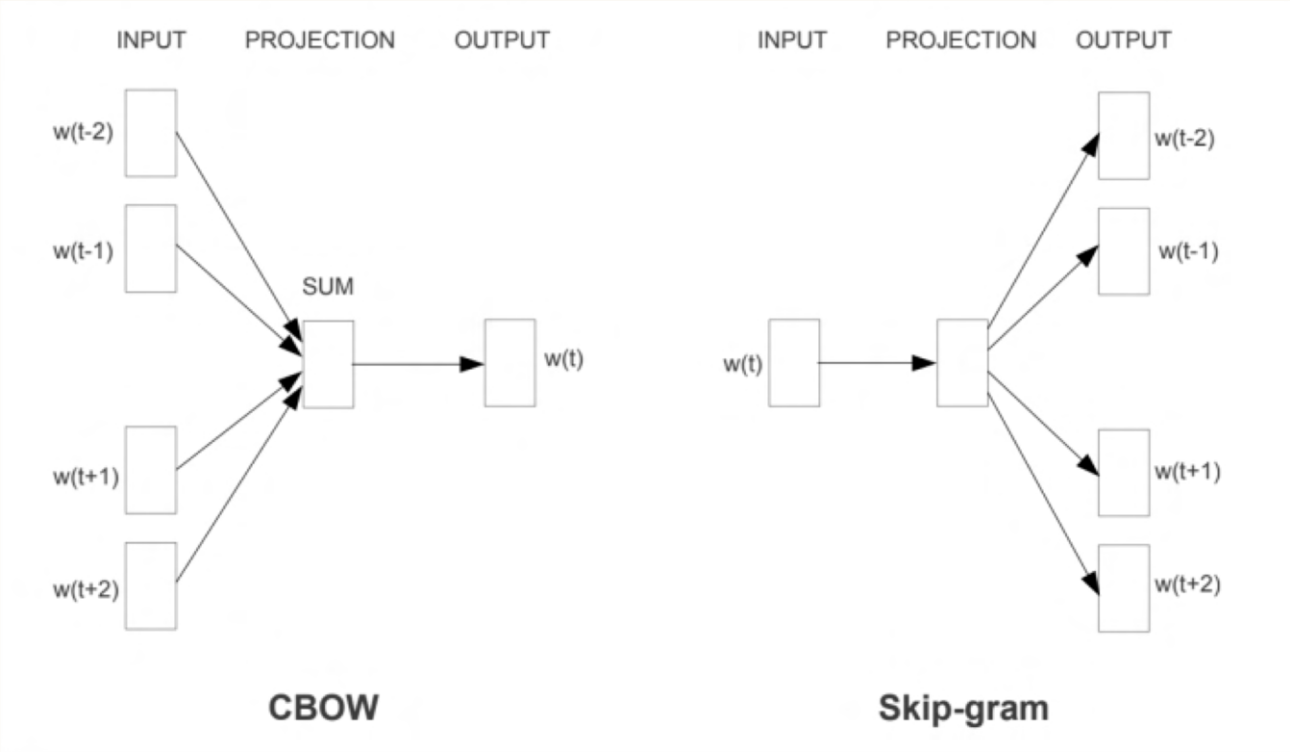
- conda create --name tutorial python=3.7
- pip install torch==1.6.0+cu101 torchvision==0.7.0+cu101 -f https://download.pytorch.org/whl/torch_stable.html → 模型訓練

※ 根據CUDA版本安裝不同版本的 pytorch
https://pytorch.org/get-started/locally/

- pip install pandas==1.3.5 → 數據操作及分析
- pip install tqdm==4.62.3 → 進度條
- pip install gensim==4.2.0 → 詞嵌入訓練

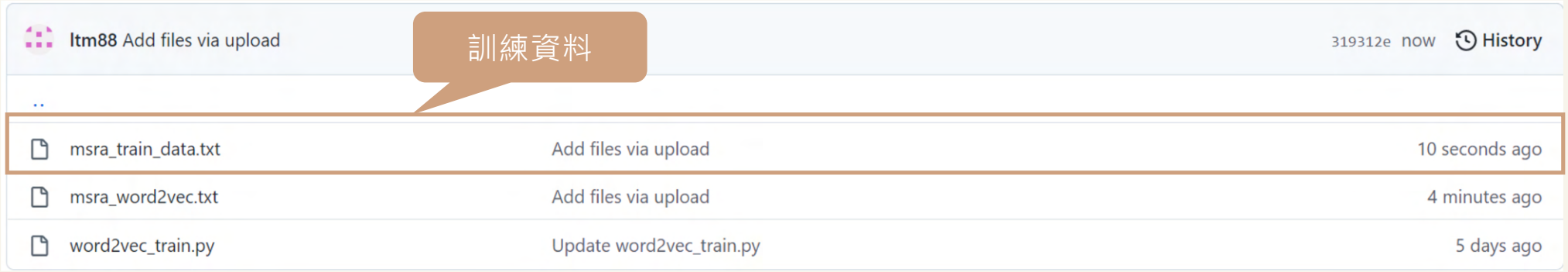
Steps 3. 詞嵌入訓練

word2vec 介紹



https://code.google.com/archive/p/word2vec/

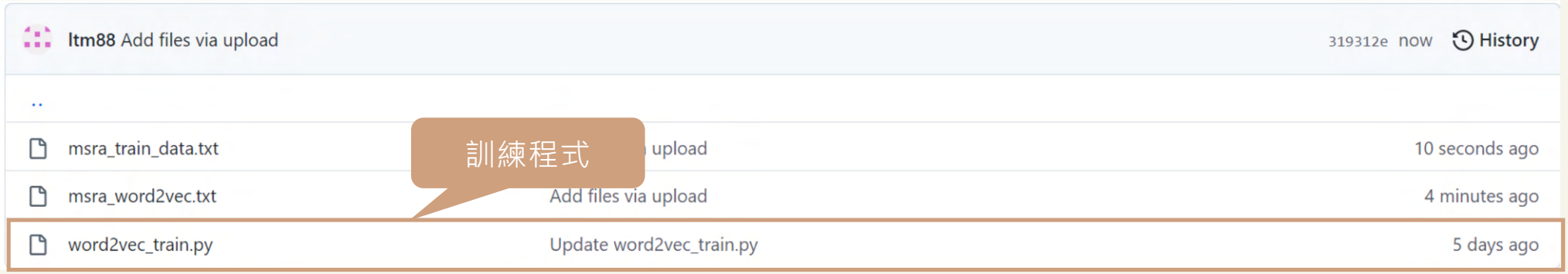
embedding 資料夾介紹



msra_train_data.txt

```
1  当希望工程救助的百万儿童成长起来，科教兴国蔚然成风时，今天有收藏价值的书你没买，明日就叫你悔不当初！
2  藏书本来就是所有传统收藏门类中的第一大户，只是我们结束温饱的时间太短而已。
3  因有关日寇在京掠夺文物详情，藏界较为重视，也是我们收藏北史料中的要件之一。
4  我们藏有一册1945年6月油印的《北文物保存保管状态之调查报告》，调查范围涉及故、历、古研所、北、北、日
   以上，洋洋三万余言，是珍贵的北史料。
5  以家乡的历史文献、特定历史时期书刊、某一名家或名著的多种出版物为专题，注意精品、非卖品、纪念品，集成系
6  我们是受到郑振铎先生、阿英先生著作的启示，从个人条件出发，瞄准现代出版史研究的空白，重点集藏解放区、
7  靠自己的鉴赏能力，将某一专题尽可能多的书籍汇集在身旁，并得到收藏界的认可，那你就是真正意义上的藏书家
8  精品、专题、系列、稀见程度才是质量的核心。
9  藏书的数量多少不能反映收藏的质量，更不是工薪层的承受范围。
10 书籍浩如烟海，靠个人的精力与财力不可能广而博之。
11 保存典籍，怡情雅好，著书立说，这才是藏书家的全部。
12 闲钱、闲地方、闲工夫固然重要，但藏书要有文化底蕴，对于书所传达出来的美感和情趣，要放到一定的专业背景
13 对于靠藏品增值来致富的说法，我们不敢非议，但从没听说过哪位藏书家靠书发财了。
14 否则，不吃不喝也得当个藏书家。
15 许多初入此道的朋友叹惜没赶上我们开始藏书时那个三五角钱买本线装书的时代。
16 近年来兴起的集藏热极大地带动了各类藏品的价格。
17 去年，我们又被评为“北首届家庭藏书状元明星户”。
```

embedding 資料夾介紹

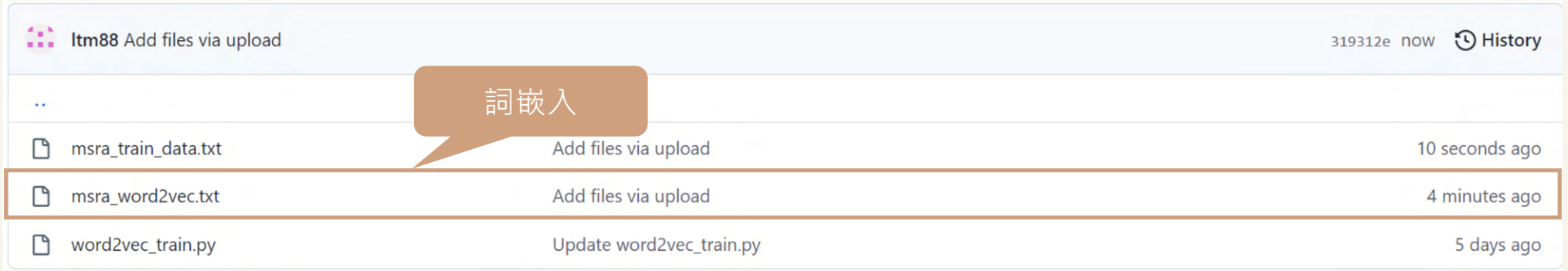


word2vec_train.py

```python
# -*- coding: utf-8 -*-

import logging

from gensim.models import word2vec

def main():

    logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
    sentences = word2vec.LineSentence("msra_train_data.txt")
    model = word2vec.Word2Vec(sentences, vector_size=300, min_count=1, epochs=10)

    #保存模型，供日後使用
    # model.save("msra_tutorial.model")
    model.wv.save_word2vec_format('msra_word2vec.txt', binary=False)
    #模型讀取方式
    # model = word2vec.Word2Vec.load("your_model_name")

if __name__ == "__main__":
    main()

```

詞嵌入維度　出現頻率　epoch

embedding 資料夾介紹



msra_word2vec.txt

總字數

```
1  4597 300
2  的 0.07178678 -0.66673344 -0.24190097 0.1872815 -0.25533 -0.43268993 0.5545681 -0.840091 0.6056946 -0.23451579 -0.013076881 0.90709156
   0.24304982 -1.1641405 0.04965666 0.4343732 0.46642032 -0.56545264 -0.6107755 0.6705009 1.0410724 -0.05998323 0.12397611 -0.012174541
   -0.99856997 0.8262848 0.79951805 0.5153007 1.111045 -0.33763522 0.57832164 0.85719466 1.3546324 -0.6544077 0.80320305 0.08723401 -0.3203125
   0.46640977 0.70025915 1.0737867 -0.31295344 0.0745044 0.03373764 -0.95569235 -0.87770027 -0.12258452 -0.44278172 0.044841576 0.50674015
   -0.608306 1.899251 -0.62762815 0.30494827 -0.9819394 -0.26518974 -0.8500857 -0.14477313 0.37207526 0.55307084 -0.97695184 0.8210223 0.9561793
   0.97675306 -0.27564883 -0.5561766 0.52587426 -1.4089365 -0.35308444 0.2787257 0.053443927 -0.16196194 -0.5106815 -0.968949 0.7962189
   -0.38950413 -1.0028594 0.9767042 -0.32198173 -0.30837008 -0.09426221 0.39134276 0.11711559 0.83293986 -0.40421343 -0.702871 -1.0654105
   0.39264333 0.05304172 -1.2207743 -0.8785136 0.10948656 -1.1478448 0.5067147 0.06667623 1.2244076 -0.18514405 0.13902922 0.12706186
   -0.57838666 -0.31363824 0.9087474 -0.56392294 -0.9019087 0.18706304 0.24259911 0.3426713 -0.1573656 0.37677777 -0.58579624 -0.10073997
   -0.047715932 0.880395 -0.35256946 -0.3335344 -0.38957974 0.63774127 0.5864072 1.3974233 0.051138002 0.43382436 0.33168322 -0.14799367
   -0.3960933 -0.5394146 0.43946147 0.9009435 -1.0161575 0.54950094 0.3622057 0.53904724 -0.79205877 -0.6827828 0.5735072 0.100954704 0.05191228
   0.6079124 0.3231929 0.48821473 0.64301324 1.6149166 0.013230636 0.574345 -0.3520925 -0.07809911 -0.09942335 -0.84003556 0.4306642 0.2194213
   0.42548025 -0.14168271 0.40751472 0.5146302 0.68805426 -0.1659402 -0.06286049 0.18110943 -0.21772897 -0.0694795 0.32798 -1.1701869 1.1630317
   0.18979178 0.40567064 0.6247707 0.3359188 -0.38614562 0.04001716 -0.42671576 -1.217215 -0.6628186 -0.34072062 0.20370205 -0.7361812
   -0.6807709 0.47184432 0.85174036 0.14314741 0.73381454 0.4726104 0.38904056 -0.6120329 0.036434386 0.41694313 1.0582484 -0.3673013
   -0.32145554 -0.06685263 0.056198582 -0.032925274 0.73679197 0.2513356 0.9733505 0.85097826 0.27613488 -0.6579208 0.37400377 0.66711503
   0.45102584 -0.93525994 0.59085876 -0.3352634 0.022064447 1.0628026 0.20754614 -0.3265849 0.19896379 1.2411556 -0.17941236 0.27808514
   -0.049582638 -0.24424842 -0.3893363 0.29918838 -0.84976107 -0.45302027 -0.17077689 1.1381761 0.05622833 0.5521833 0.20695488 -1.2354794
   0.6779572 1.9593011 -0.18822397 -0.05269354 -0.3540321 0.81638163 -0.3259991 0.47610345 0.19568081 -0.9785319 -0.7300356 0.15698427
   -0.13344677 -0.15481272 -0.030200899 0.009212236 0.50759155 0.08739441 -1.1607878 -0.013080179 -0.24200979 -0.46228552 0.26831722 0.33612815
   0.46466225 -0.71556365 0.09187978 -0.12648182 -0.30961356 0.45455083 -0.2450905 -0.0066274623 -0.23197371 -0.5604785 0.52927893 0.33792013
   0.4763087 -0.3775083 0.60251564 0.100057684 0.13694783 0.5749381 -0.8557217 -0.12723477 0.5507165 0.586475 -0.22478816 0.15446551 0.5364823
   0.016301285 -0.49955377 0.004319039 0.7215278 0.58162934 0.108547054 -0.8893235 0.03828856 -0.2351299 0.27106532 -0.293951 0.39948493
   -0.5105325 -0.22952692 -0.34071004 -0.0017946797 -0.26452723 0.2269461 0.20143875 -0.37920043 0.69019 -0.09258967 -0.6194667 -0.997748
   0.044556532 -0.79035044 0.17834833 -0.5016108 0.2759766 0.19204079
```

msra_word2vec.txt

向量維度

```
1  4597 300
2  的 0.07178678 -0.66673344 -0.24190097 0.1872815 -0.25533 -0.43268993 0.5545681 -0.840091 0.6056946 -0.23451579 -0.013076881 0.90709156
   0.24304982 -1.1641405 0.04965666 0.4343732 0.46642032 -0.56545264 -0.6107755 0.6705009 1.0410724 -0.05998323 0.12397611 -0.012174541
   -0.99856997 0.8262848 0.79951805 0.5153007 1.111045 -0.33763522 0.57832164 0.85719466 1.3546324 -0.6544077 0.80320305 0.08723401 -0.3203125
   0.46640977 0.70025915 1.0737867 -0.31295344 0.0745044 0.03373764 -0.95569235 -0.87770027 -0.12258452 -0.44278172 0.044841576 0.50674015
   -0.608306 1.899251 -0.62762815 0.30494827 -0.9819394 -0.26518974 -0.8500857 -0.14477313 0.37207526 0.55307084 -0.97695184 0.8210223 0.9561793
   0.97675306 -0.27564883 -0.5561766 0.52587426 -1.4089365 -0.35308444 0.2787257 0.053443927 -0.16196194 -0.5106815 -0.968949 0.7962189
   -0.38950413 -1.0028594 0.9767042 -0.32198173 -0.30837008 -0.09426221 0.39134276 0.11711559 0.83293986 -0.40421343 -0.702871 -1.0654105
   0.39264333 0.05304172 -1.2207743 -0.8785136 0.10948656 -1.1478448 0.5067147 0.06667623 1.2244076 -0.18514405 0.13902922 0.12706186
   -0.57838666 -0.31363824 0.9087474 -0.56392294 -0.9019087 0.18706304 0.24259911 0.3426713 -0.1573656 0.37677777 -0.58579624 -0.10073997
   -0.047715932 0.880395 -0.35256946 -0.3335344 -0.38957974 0.63774127 0.5864072 1.3974233 0.051138002 0.43382436 0.33168322 -0.14799367
   -0.3960933 -0.5394146 0.43946147 0.9009435 -1.0161575 0.54950094 0.3622057 0.53904724 -0.79205877 -0.6827828 0.5735072 0.100954704 0.05191228
   0.6079124 0.3231929 0.48821473 0.64301324 1.6149166 0.013230636 0.574345 -0.3520925 -0.07809911 -0.09942335 -0.84003556 0.4306642 0.2194213
   0.42548025 -0.14168271 0.40751472 0.5146302 0.68805426 -0.1659402 -0.06286049 0.18110943 -0.21772897 -0.0694795 0.32798 -1.1701869 1.1630317
   0.18979178 0.40567064 0.6247707 0.3359188 -0.38614562 0.04001716 -0.42671576 -1.217215 -0.6628186 -0.34072062 0.20370205 -0.7361812
   -0.6807709 0.47184432 0.85174036 0.14314741 0.73381454 0.4726104 0.38904056 -0.6120329 0.036434386 0.41694313 1.0582484 -0.3673013
   -0.32145554 -0.06685263 0.056198582 -0.032925274 0.73679197 0.2513356 0.9733505 0.85097826 0.27613488 -0.6579208 0.37400377 0.66711503
   0.45102584 -0.93525994 0.59085876 -0.3352634 0.022064447 1.0628026 0.20754614 -0.3265849 0.19896379 1.2411556 -0.17941236 0.27808514
   -0.049582638 -0.24424842 -0.3893363 0.29918838 -0.84976107 -0.45302027 -0.17077689 1.1381761 0.05622833 0.5521833 0.20695488 -1.2354794
   0.6779572 1.9593011 -0.18822397 -0.05269354 -0.3540321 0.81638163 -0.3259991 0.47610345 0.19568081 -0.9785319 -0.7300356 0.15698427
   -0.13344677 -0.15481272 -0.030200899 0.009212236 0.50759155 0.08739441 -1.1607878 -0.013080179 -0.24200979 -0.46228552 0.26831722 0.33612815
   0.46466225 -0.71556365 0.09187978 -0.12648182 -0.30961356 0.45455083 -0.2450905 -0.0066274623 -0.23197371 -0.5604785 0.52927893 0.33792013
   0.4763087 -0.3775083 0.60251564 0.100057684 0.13694783 0.5749381 -0.8557217 -0.12723477 0.5507165 0.586475 -0.22478816 0.15446551 0.5364823
   0.016301285 -0.49955377 0.004319039 0.7215278 0.58162934 0.108547054 -0.8893235 0.03828856 -0.2351299 0.27106532 -0.293951 0.39948493
   -0.5105325 -0.22952692 -0.34071004 -0.0017946797 -0.26452723 0.2269461 0.20143875 -0.37920043 0.69019 -0.09258967 -0.6194667 -0.997748
   0.044556532 -0.79035044 0.17834833 -0.5016108 0.2759766 0.19204079
```

msra_word2vec.txt

字元

```
1  4597 300
2  的 0.07178678 -0.66673344 -0.24190097 0.1872815 -0.25533 -0.43268993 0.5545681 -0.840091 0.6056946 -0.23451579 -0.013076881 0.90709156
   0.24304982 -1.1641405 0.04965666 0.4343732 0.46642032 -0.56545264 -0.6107755 0.6705009 1.0410724 -0.05998323 0.12397611 -0.012174541
   -0.99856997 0.8262848 0.79951805 0.5153007 1.111045 -0.33763522 0.57832164 0.85719466 1.3546324 -0.6544077 0.80320305 0.08723401 -0.3203125
   0.46640977 0.70025915 1.0737867 -0.31295344 0.0745044 0.03373764 -0.95569235 -0.87770027 -0.12258452 -0.44278172 0.044841576 0.50674015
   -0.608306 1.899251 -0.62762815 0.30494827 -0.9819394 -0.26518974 -0.8500857 -0.14477313 0.37207526 0.55307084 -0.97695184 0.8210223 0.9561793
   0.97675306 -0.27564883 -0.5561766 0.52587426 -1.4089365 -0.35308444 0.2787257 0.053443927 -0.16196194 -0.5106815 -0.968949 0.7962189
   -0.38950413 -1.0028594 0.9767042 -0.32198173 -0.30837008 -0.09426221 0.39134276 0.11711559 0.83293986 -0.40421343 -0.702871 -1.0654105
   0.39264333 0.05304172 -1.2207743 -0.8785136 0.10948656 -1.1478448 0.5067147 0.06667623 1.2244076 -0.18514405 0.13902922 0.12706186
   -0.57838666 -0.31363824 0.9087474 -0.56392294 -0.9019087 0.18706304 0.24259911 0.3426713 -0.1573656 0.37677777 -0.58579624 -0.10073997
   -0.047715932 0.880395 -0.35256946 -0.3335344 -0.38957974 0.63774127 0.5864072 1.3974233 0.051138002 0.43382436 0.33168322 -0.14799367
   -0.3960933 -0.5394146 0.43946147 0.9009435 -1.0161575 0.54950094 0.3622057 0.53904724 -0.79205877 -0.6827828 0.5735072 0.100954704 0.05191228
   0.6079124 0.3231929 0.48821473 0.64301324 1.6149166 0.013230636 0.574345 -0.3520925 -0.07809911 -0.09942335 -0.84003556 0.4306642 0.2194213
   0.42548025 -0.14168271 0.40751472 0.5146302 0.68805426 -0.1659402 -0.06286049 0.18110943 -0.21772897 -0.0694795 0.32798 -1.1701869 1.1630317
   0.18979178 0.40567064 0.6247707 0.3359188 -0.38614562 0.04001716 -0.42671576 -1.217215 -0.6628186 -0.34072062 0.20370205 -0.7361812
   -0.6807709 0.47184432 0.85174036 0.14314741 0.73381454 0.4726104 0.38904056 -0.6120329 0.036434386 0.41694313 1.0582484 -0.3673013
   -0.32145554 -0.06685263 0.056198582 -0.032925274 0.73679197 0.2513356 0.9733505 0.85097826 0.27613488 -0.6579208 0.37400377 0.66711503
   0.45102584 -0.93525994 0.59085876 -0.3352634 0.022064447 1.0628026 0.20754614 -0.3265849 0.19896379 1.2411556 -0.17941236 0.27808514
   -0.049582638 -0.24424842 -0.3893363 0.29918838 -0.84976107 -0.45302027 -0.17077689 1.1381761 0.05622833 0.5521833 0.20695488 -1.2354794
   0.6779572 1.9593011 -0.18822397 -0.05269354 -0.3540321 0.81638163 -0.3259991 0.47610345 0.19568081 -0.9785319 -0.7300356 0.15698427
   -0.13344677 -0.15481272 -0.030200899 0.009212236 0.50759155 0.08739441 -1.1607878 -0.013080179 -0.24200979 -0.46228552 0.26831722 0.33612815
   0.46466225 -0.71556365 0.09187978 -0.12648182 -0.30961356 0.45455083 -0.2450905 -0.0066274623 -0.23197371 -0.5604785 0.52927893 0.33792013
   0.4763087 -0.3775083 0.60251564 0.100057684 0.13694783 0.5749381 -0.8557217 -0.12723477 0.5507165 0.586475 -0.22478816 0.15446551 0.5364823
   0.016301285 -0.49955377 0.004319039 0.7215278 0.58162934 0.108547054 -0.8893235 0.03828856 -0.2351299 0.27106532 -0.293951 0.39948493
   -0.5105325 -0.22952692 -0.34071004 -0.0017946797 -0.26452723 0.2269461 0.20143875 -0.37920043 0.69019 -0.09258967 -0.6194667 -0.997748
   0.044556532 -0.79035044 0.17834833 -0.5016108 0.2759766 0.19204079
```

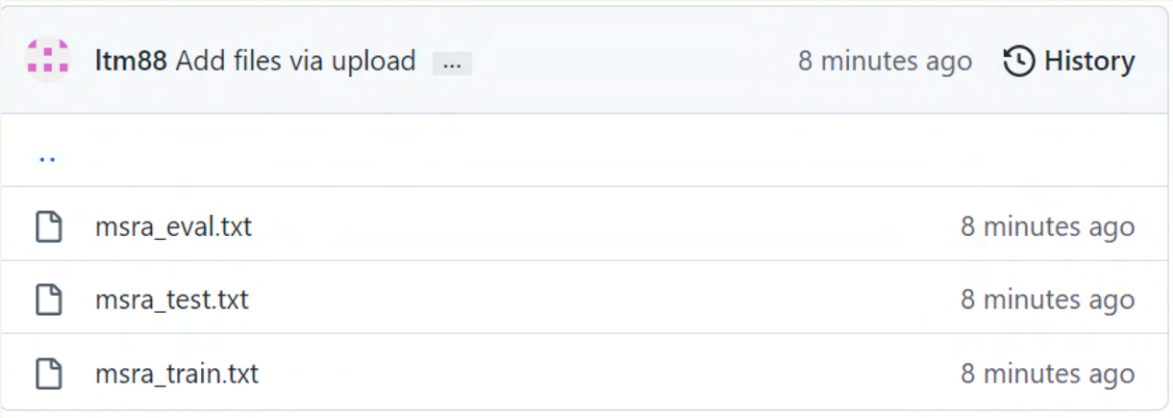
msra_word2vec.txt

300維向量

```
4597 300
的 0.07178678 -0.66673344 -0.24190097 0.1872815 -0.25533 -0.43268993 0.5545681 -0.840091 0.6056946 -0.23451579 -0.013076881 0.90709156
0.24304982 -1.1641405 0.04965666 0.4343732 0.46642032 -0.56545264 -0.6107755 0.6705009 1.0410724 -0.05998323 0.12397611 -0.012174541
-0.99856997 0.8262848 0.79951805 0.5153007 1.111045 -0.33763522 0.57832164 0.85719466 1.3546324 -0.6544077 0.80320305 0.08723401 -0.3203125
0.46640977 0.70025915 1.0737867 -0.31295344 0.0745044 0.03373764 -0.95569235 -0.87770027 -0.12258452 -0.44278172 0.044841576 0.50674015
-0.608306 1.899251 -0.62762815 0.30494827 -0.9819394 -0.26518974 -0.8500857 -0.14477313 0.37207526 0.55307084 -0.97695184 0.8210223 0.9561793
0.97675306 -0.27564883 -0.5561766 0.52587426 -1.4089365 -0.35308444 0.2787257 0.053443927 -0.16196194 -0.5106815 -0.968949 0.7962189
-0.38950413 -1.0028594 0.9767042 -0.32198173 -0.30837008 -0.09426221 0.39134276 0.11711559 0.83293986 -0.40421343 -0.702871 -1.0654105
0.39264333 0.05304172 -1.2207743 -0.8785136 0.10948656 -1.1478448 0.5067147 0.06667623 1.2244076 -0.18514405 0.13902922 0.12706186
-0.57838666 -0.31363824 0.9087474 -0.56392294 -0.9019087 0.18706304 0.24259911 0.3426713 -0.1573656 0.37677777 -0.58579624 -0.10073997
-0.047715932 0.880395 -0.35256946 -0.3335344 -0.38957974 0.63774127 0.5864072 1.3974233 0.051138002 0.43382436 0.33168322 -0.14799367
-0.3960933 -0.5394146 0.43946147 0.9009435 -1.0161575 0.54950094 0.3622057 0.53904724 -0.79205877 -0.6827828 0.5735072 0.100954704 0.05191228
0.6079124 0.3231929 0.48821473 0.64301324 1.6149166 0.013230636 0.574345 -0.3520925 -0.07809911 -0.09942335 -0.84003556 0.4306642 0.2194213
0.42548025 -0.14168271 0.40751472 0.5146302 0.68805426 -0.1659402 -0.06286049 0.18110943 -0.21772897 -0.0694795 0.32798 -1.1701869 1.1630317
0.18979178 0.40567064 0.6247707 0.3359188 -0.38614562 0.04001716 -0.42671576 -1.217215 -0.6628186 -0.34072062 0.20370205 -0.7361812
-0.6807709 0.47184432 0.85174036 0.14314741 0.73381454 0.4726104 0.38904056 -0.6120329 0.036434386 0.41694313 1.0582484 -0.3673013
-0.32145554 -0.06685263 0.056198582 -0.032925274 0.73679197 0.2513356 0.9733505 0.85097826 0.27613488 -0.6579208 0.37400377 0.66711503
0.45102584 -0.93525994 0.59085876 -0.3352634 0.022064447 1.0628026 0.20754614 -0.3265849 0.19896379 1.2411556 -0.17941236 0.27808514
-0.049582638 -0.24424842 -0.3893363 0.29918838 -0.84976107 -0.45302027 -0.17077689 1.1381761 0.05622833 0.5521833 0.20695488 -1.2354794
0.6779572 1.9593011 -0.18822397 -0.05269354 -0.3540321 0.81638163 -0.3259991 0.47610345 0.19568081 -0.9785319 -0.7300356 0.15698427
-0.13344677 -0.15481272 -0.030200899 0.009212236 0.50759155 0.08739441 -1.1607878 -0.013080179 -0.24200979 -0.46228552 0.26831722 0.33612815
0.46466225 -0.71556365 0.09187978 -0.12648182 -0.30961356 0.45455083 -0.2450905 -0.0066274623 -0.23197371 -0.5604785 0.52927893 0.33792013
0.4763087 -0.3775083 0.60251564 0.100057684 0.13694783 0.5749381 -0.8557217 -0.12723477 0.5507165 0.586475 -0.22478816 0.15446551 0.5364823
0.016301285 -0.49955377 0.004319039 0.7215278 0.58162934 0.108547054 -0.8893235 0.03828856 -0.2351299 0.27106532 -0.293951 0.39948493
-0.5105325 -0.22952692 -0.34071004 -0.0017946797 -0.26452723 0.2269461 0.20143875 -0.37920043 0.69019 -0.09258967 -0.6194667 -0.997748
0.044556532 -0.79035044 0.17834833 -0.5016108 0.2759766 0.19204079
```

Steps 4. BiLSTM+CRF 訓練

訓練資料



	句子數量	標籤總量	PER	LOC	ORG
train	33,887	58,738	13,508	28,946	16,284
eval	9,671	13,274	3,580	5,784	3,910
test	4,838	6,749	1,851	3,264	1,634

msra_train.txt
msra_eval.txt

```
因 O
有 O
关 O
日 B-LOC
寇 O
在 O
京 B-LOC
掠 O
夺 O
文 O
物 O
详 O
情 O
， O
藏 O
界 O
较 O
为 O
重 O
视 O
， O
```

msra_test.txt

```
既
讲
“
写
字
”
：
先
须
横
平
竖
直
，
端
正
匀
齐
，
```

main.py – 設定參數

python main.py --mode Train --save_model_name tutorial.pt --Epoch 1

```
parser = argparse.ArgumentParser()

#使用模式
parser.add_argument("--mode",default="Train",type=str,help="Train or Test")
#預存檔在 save_model_dir_path 中 模型參數的名稱
parser.add_argument("--save_model_name",default="tutorial.pt",type=str,help="存擋 save_model_dir_path 中 模型參數的名稱(.pt格式)")
#predict file name
parser.add_argument("--predict_name",default="predict.txt",type=str,help="存擋 predict 檔案名稱(.txt格式)")
#讀取 saved_model 裡模型參數的完整路徑
parser.add_argument("--load_model_name",default="tutorial",type=str,help="讀取的model name")
#設定 Training data path
parser.add_argument("--Train_data_path",default="data/msra_train.txt",type=str,help="設定 Training data path")
#設定 Evaluation data path
parser.add_argument("--Eval_data_path",default="data/msra_dev.txt",type=str,help="設定 Evaluation data path")
#設定 Testing data path
parser.add_argument("--Test_data_path",default="data/msra_test.txt",type=str,help="設定 Testing data path")
#設定 Training Epoch
parser.add_argument("--Epoch",default=40, type=int,help="設定 Training Epoch")
#設定 learning rate
parser.add_argument("--lr",default=4e-3,type=float,help="設定 learning rate")
#設定 batch size
parser.add_argument("--batch_size",default=256,type=int,help="設定 batch size")
#設定 lstm hidden dim
parser.add_argument("--lstm_hidden_dim",default=1024,type=int,help="設定 lstm hidden dim")
#設定 lstm dropout rate
parser.add_argument("--lstm_dropout_rate",default=0.1, type=int,help="設定 lstm dropout rate")
# 設定Random seed
parser.add_argument("--seed",default=87,type=int,help="random seed set")
#選擇gpu
parser.add_argument("--gpu",default='0',type=str,help="which gpu")
#pretrai nembedding
parser.add_argument("--embedding",default='embedding/msra_word2vec.txt',type=str,help="which gpu")
#embedding dimesion
parser.add_argument("--dimesion",default=300,type=int,help="embedding dimension")
args = parser.parse_args()
args = vars(args)
```

main.py – 讀取參數

```
51  #設定gpu
52  os.environ['CUDA_VISIBLE_DEVICES'] = args['gpu']  # 指定GPU
53  device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
54
55  # 設定隨機種子值，以確保輸出是確定的
56  seed_val = args['seed']
57  random.seed(seed_val)
58  np.random.seed(seed_val)
59  torch.manual_seed(seed_val)
60  torch.cuda.manual_seed_all(seed_val)
61
62  # Train or Test or Predict
63  mode = args['mode']
64
65  #hyper parameter
66  EPOCH = args['Epoch']
67  lr = args['lr']
68  batch_size = args['batch_size']
69  lstm_hidden_dim = args['lstm_hidden_dim']
70  lstm_dropout_rate = args['lstm_dropout_rate']
71  dimesion=args['dimesion']
72  roberta_freeze = True
73  word2vec = pd.read_csv(args['embedding'], sep=" ", quoting=3, header=None, index_col=0,skiprows=1)
74  word2vec_embedding = {key: val.values for key, val in word2vec.T.items()}
75
```

設定GPU

main.py – 讀取參數

```
51  #設定gpu
52  os.environ['CUDA_VISIBLE_DEVICES'] = args['gpu']  # 指定GPU
53  device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
54
55  # 設定隨機種子值，以確保輸出是確定的
56  seed_val = args['seed']
57  random.seed(seed_val)
58  np.random.seed(seed_val)
59  torch.manual_seed(seed_val)
60  torch.cuda.manual_seed_all(seed_val)
61
62  # Train or Test or Predict
63  mode = args['mode']
64
65  #hyper parameter
66  EPOCH = args['Epoch']
67  lr = args['lr']
68  batch_size = args['batch_size']
69  lstm_hidden_dim = args['lstm_hidden_dim']
70  lstm_dropout_rate = args['lstm_dropout_rate']
71  dimesion=args['dimesion']
72  roberta_freeze = True
73  word2vec = pd.read_csv(args['embedding'], sep=" ", quoting=3, header=None, index_col=0,skiprows=1)
74  word2vec_embedding = {key: val.values for key, val in word2vec.T.items()}
75
```

設定隨機種子值

main.py – 讀取參數

```
51  #設定gpu
52  os.environ['CUDA_VISIBLE_DEVICES'] = args['gpu']  # 指定GPU
53  device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
54
55  # 設定隨機種子值，以確保輸出是確定的
56  seed_val = args['seed']
57  random.seed(seed_val)
58  np.random.seed(seed_val)
59  torch.manual_seed(seed_val)
60  torch.cuda.manual_seed_all(seed_val)
61
62  # Train or Test or Predict
63  mode = args['mode']
64
65  #hyper parameter
66  EPOCH = args['Epoch']
67  lr = args['lr']
68  batch_size = args['batch_size']
69  lstm_hidden_dim = args['lstm_hidden_dim']
70  lstm_dropout_rate = args['lstm_dropout_rate']
71  dimesion=args['dimesion']
72  roberta_freeze = True
73  word2vec = pd.read_csv(args['embedding'], sep=" ", quoting=3, header=None, index_col=0,skiprows=1)
74  word2vec_embedding = {key: val.values for key, val in word2vec.T.items()}
75
```

讀取超參數

main.py – 路徑設定

```python
#Training Mode
if mode == 'Train':

    #create dir for save model
    assert len(args['save_model_name'].split('.pt')) == 2 ,'模型名稱錯誤:{xxx.pt}'
    dir_name = args['save_model_name'].split('.pt')[0]

    model_path = os.path.join('saved_model',dir_name)
    if not os.path.isdir(model_path):
        os.mkdir(model_path)

    #save parameter
    with open(os.path.join(model_path,'parameter.txt'),'w',encoding='utf-8') as p_file:
        for item in args:
            p_file.write(f"{item}: {args[item]}\n")

    #Load Training data
    print("Loading Training data")
    Train_data_path = args['Train_data_path']
    dataset_train = Dataset(Train_data_path,word2vec,dimesion,mode='Train')
    dataloader_train = DataLoader(dataset_train, collate_fn=dataset_train.collate_fn,
                                  batch_size=batch_size, shuffle=False, drop_last=True)

    print(f"training data size : {dataset_train.__len__()}\n")

    #Load Eval Data
    print("Loading Evaluation data")
    Eval_data_path = args['Eval_data_path']
    dataset_eval = Dataset(Eval_data_path,word2vec,dimesion,mode='Eval')
    dataloader_eval = DataLoader(dataset_eval, collate_fn=dataset_eval.collate_fn,
                                  batch_size=batch_size, shuffle=False, drop_last=False)

    print(f"evaluation data size : {dataset_eval.__len__()} \n")

    #Create Model
    model = BiLSTM_CRF(len(dataset_train.id2tag),lstm_hidden_dim=lstm_hidden_dim,lstm_dropout_rate=lstm_dropout_rate).to(device)
    model.reset_parameters()

    #Create optimizer
    optimizer = Adam(filter(lambda p: p.requires_grad, model.parameters()), lr=lr)
```

儲存模型路徑

main.py – 儲存超參數

```python
#Training Mode
if mode == 'Train':

    #create dir for save model
    assert len(args['save_model_name'].split('.pt')) == 2 ,'模型名稱錯誤:{xxx.pt}'
    dir_name = args['save_model_name'].split('.pt')[0]

    model_path = os.path.join('saved_model',dir_name)
    if not os.path.isdir(model_path):
        os.mkdir(model_path)

    #save parameter
    with open(os.path.join(model_path,'parameter.txt'),'w',encoding='utf-8') as p_file:
        for item in args:
            p_file.write(f"{item}: {args[item]}\n")

    #Load Training data
    print("Loading Training data")
    Train_data_path = args['Train_data_path']
    dataset_train = Dataset(Train_data_path,word2vec,dimesion,mode='Train')
    dataloader_train = DataLoader(dataset_train, collate_fn=dataset_train.collate_fn,
                                  batch_size=batch_size, shuffle=False, drop_last=True)

    print(f"training data size : {dataset_train.__len__()}\n")

    #Load Eval Data
    print("Loading Evaluation data")
    Eval_data_path = args['Eval_data_path']
    dataset_eval = Dataset(Eval_data_path,word2vec,dimesion,mode='Eval')
    dataloader_eval = DataLoader(dataset_eval, collate_fn=dataset_eval.collate_fn,
                                  batch_size=batch_size, shuffle=False, drop_last=False)

    print(f"evaluation data size : {dataset_eval.__len__()} \n")

    #Create Model
    model = BiLSTM_CRF(len(dataset_train.id2tag),lstm_hidden_dim=lstm_hidden_dim,lstm_dropout_rate=lstm_dropout_rate).to(device)
    model.reset_parameters()

    #Create optimizer
    optimizer = Adam(filter(lambda p: p.requires_grad, model.parameters()), lr=lr)
```

儲存超參數

main.py – 讀取資料

```python
#Training Mode
if mode == 'Train':

    #create dir for save model
    assert len(args['save_model_name'].split('.pt')) == 2 ,'模型名稱錯誤:{xxx.pt}'
    dir_name = args['save_model_name'].split('.pt')[0]

    model_path = os.path.join('saved_model',dir_name)
    if not os.path.isdir(model_path):
        os.mkdir(model_path)

    #save parameter
    with open(os.path.join(model_path,'parameter.txt'),'w',encoding='utf-8') as p_file:
        for item in args:
            p_file.write(f"{item}: {args[item]}\n")

    #Load Training data
    print("Loading Training data")
    Train_data_path = args['Train_data_path']
    dataset_train = Dataset(Train_data_path,word2vec,dimesion,mode='Train')
    dataloader_train = DataLoader(dataset_train, collate_fn=dataset_train.collate_fn,
                                  batch_size=batch_size, shuffle=False, drop_last=True)

    print(f"training data size : {dataset_train.__len__()}\n")

    #Load Eval Data
    print("Loading Evaluation data")
    Eval_data_path = args['Eval_data_path']
    dataset_eval = Dataset(Eval_data_path,word2vec,dimesion,mode='Eval')
    dataloader_eval = DataLoader(dataset_eval, collate_fn=dataset_eval.collate_fn,
                                  batch_size=batch_size, shuffle=False, drop_last=False)

    print(f"evaluation data size : {dataset_eval.__len__()} \n")

    #Create Model
    model = BiLSTM_CRF(len(dataset_train.id2tag),lstm_hidden_dim=lstm_hidden_dim,lstm_dropout_rate=lstm_dropout_rate).to(device)
    model.reset_parameters()

    #Create optimizer
    optimizer = Adam(filter(lambda p: p.requires_grad, model.parameters()), lr=lr)
```

讀取資料

package/dataset.py

```python
class Dataset(Dataset):

    def __init__(self,data_path,embedding_path,dimesion,mode='Train'):
        self.mode = mode
        self.data = read_csvdata(data_path,mode=self.mode)
        self.maxlen = 512
        self.embedding = {key: val.values for key, val in embedding_path.T.items()}
        self.dimesion=dimesion
        if  self.mode=='Train':
            self.tag2id = self.auto_get_tag2id()
            self.id2tag = self.auto_get_id2tag(self.tag2id)
            self.save_json()

        elif  self.mode=='Eval' or  self.mode=='Test':
            self.tag2id,self.id2tag = self.load_json()

        print(self.mode)
        print(f"tag2id: {self.tag2id}")
        print(f"id2tag: {self.id2tag}")
        print(f"tag number: {len(self.tag2id)} \n")
```

讀取資料
及詞嵌入

package/dataset.py

```python
class Dataset(Dataset):

    def __init__(self,data_path,embedding_path,dimesion,mode='Train'):
        self.mode = mode
        self.data = read_csvdata(data_path,mode=self.mode)
        self.maxlen = 512
        self.embedding = {key: val.values for key, val in embedding_path.T.items()}
        self.dimesion=dimesion
        if  self.mode=='Train':
            self.tag2id = self.auto_get_tag2id()
            self.id2tag = self.auto_get_id2tag(self.tag2id)
            self.save_json()

        elif  self.mode=='Eval' or  self.mode=='Test':
            self.tag2id,self.id2tag = self.load_json()

        print(self.mode)
        print(f"tag2id: {self.tag2id}")
        print(f"id2tag: {self.id2tag}")
        print(f"tag number: {len(self.tag2id)} \n")
```

標籤轉換

data/tag2id.json

```
1  {"O": 0, "B-LOC": 1, "I-LOC": 2, "B-ORG": 3, "I-ORG": 4, "B-PER": 5, "I-PER": 6}
2
3
```

package/dataset.py

```python
def collate_fn(self, batch):

    def create_embedding_matrix(word_index,embedding_dict,dimension):
        embedding_matrix=np.zeros((512,dimension))
        for index,word in enumerate(word_index):
            if word in embedding_dict:
                embedding_matrix[index]=embedding_dict[word]
        return embedding_matrix

    label_list,mask_list=list(),list()
    embedding=torch.tensor(float('nan'))

    if self.mode=='Train' or self.mode == 'Eval':
        for char_list,label in batch:
            embedding_matrix=create_embedding_matrix(char_list,embedding_dict=self.embedding,dimension=self.dimesion)
            embedding_matrix = torch.tensor(embedding_matrix)
            if torch.all(torch.isfinite(embedding))== False:
                embedding=embedding_matrix.view(1,512,self.dimesion)
            else:
                embedding=torch.cat((embedding, embedding_matrix.view(1,512,self.dimesion)), 0)

            label_spec_list =label
            label_spec_pad_list = label_spec_list + [0 for _ in range(self.maxlen - len( char_list ) )]
            label_list.append(label_spec_pad_list)

            mask_1 = [1 for _ in range(len(char_list))]
            mask_0 = [0 for _ in range(self.maxlen-( len(char_list)))]
            attention_mask = mask_1 + mask_0
            mask_list.append(attention_mask)

        label_tensor = torch.tensor(label_list)
        mask_tensor = torch.tensor(mask_list,dtype=torch.uint8)
        embedding_tensor=torch.tensor(embedding)

        return embedding_tensor,mask_tensor,label_tensor
```

轉換成詞嵌入

package/dataset.py

```
66      def collate_fn(self, batch):
67
68          def create_embedding_matrix(word_index,embedding_dict,dimension):
69              embedding_matrix=np.zeros((512,dimension))
70              for index,word in enumerate(word_index):
71                  if word in embedding_dict:
72                      embedding_matrix[index]=embedding_dict[word]
73              return embedding_matrix
74
75          label_list,mask_list=list(),list()
76          embedding=torch.tensor(float('nan'))
77
78          if self.mode=='Train' or self.mode == 'Eval':
79              for char_list,label in batch:
80                  embedding_matrix=create_embedding_matrix(char_list,embedding_dict=self.embedding,dimension=self.dimesion)
81                  embedding_matrix = torch.tensor(embedding_matrix)
82                  if torch.all(torch.isfinite(embedding))== False:
83                      embedding=embedding_matrix.view(1,512,self.dimesion)
84                  else:
85                      embedding=torch.cat((embedding, embedding_matrix.view(1,512,self.dimesion)), 0)
86
87                  label_spec_list =label
88                  label_spec_pad_list = label_spec_list + [0 for _ in range(self.maxlen - len( char_list ) )]
89                  label_list.append(label_spec_pad_list)
90
91                  mask_1 = [1 for _ in range(len(char_list))]
92                  mask_0 = [0 for _ in range(self.maxlen-( len(char_list)))]
93                  attention_mask = mask_1 + mask_0
94                  mask_list.append(attention_mask)
95
96              label_tensor = torch.tensor(label_list)
97              mask_tensor = torch.tensor(mask_list,dtype=torch.uint8)
98              embedding_tensor=torch.tensor(embedding)
99
100             return embedding_tensor,mask_tensor,label_tensor
```

儲存標籤

package/dataset.py

```python
def collate_fn(self, batch):

    def create_embedding_matrix(word_index,embedding_dict,dimension):
        embedding_matrix=np.zeros((512,dimension))
        for index,word in enumerate(word_index):
            if word in embedding_dict:
                embedding_matrix[index]=embedding_dict[word]
        return embedding_matrix

    label_list,mask_list=list(),list()
    embedding=torch.tensor(float('nan'))

    if self.mode=='Train' or self.mode == 'Eval':
        for char_list,label in batch:
            embedding_matrix=create_embedding_matrix(char_list,embedding_dict=self.embedding,dimension=self.dimesion)
            embedding_matrix = torch.tensor(embedding_matrix)
            if torch.all(torch.isfinite(embedding))== False:
                embedding=embedding_matrix.view(1,512,self.dimesion)
            else:
                embedding=torch.cat((embedding, embedding_matrix.view(1,512,self.dimesion)), 0)

            label_spec_list =label
            label_spec_pad_list = label_spec_list + [0 for _ in range(self.maxlen - len( char_list ) )]
            label_list.append(label_spec_pad_list)

            mask_1 = [1 for _ in range(len(char_list))]
            mask_0 = [0 for _ in range(self.maxlen-( len(char_list)))]
            attention_mask = mask_1 + mask_0
            mask_list.append(attention_mask)

        label_tensor = torch.tensor(label_list)
        mask_tensor = torch.tensor(mask_list,dtype=torch.uint8)
        embedding_tensor=torch.tensor(embedding)

        return embedding_tensor,mask_tensor,label_tensor
```

建立遮罩

package/dataset.py

```
66      def collate_fn(self, batch):
67
68          def create_embedding_matrix(word_index,embedding_dict,dimension):
69              embedding_matrix=np.zeros((512,dimension))
70              for index,word in enumerate(word_index):
71                  if word in embedding_dict:
72                      embedding_matrix[index]=embedding_dict[word]
73              return embedding_matrix
74
75          label_list,mask_list=list(),list()
76          embedding=torch.tensor(float('nan'))
77
78          if self.mode=='Train' or self.mode == 'Eval':
79              for char_list,label in batch:
80                  embedding_matrix=create_embedding_matrix(char_list,embedding_dict=self.embedding,dimension=self.dimesion)
81                  embedding_matrix = torch.tensor(embedding_matrix)
82                  if torch.all(torch.isfinite(embedding))== False:
83                      embedding=embedding_matrix.view(1,512,self.dimesion)
84                  else:
85                      embedding=torch.cat((embedding, embedding_matrix.view(1,512,self.dimesion)), 0)
86
87                  label_spec_list =label
88                  label_spec_pad_list = label_spec_list + [0 for _ in range(self.maxlen - len( char_list ) )]
89                  label_list.append(label_spec_pad_list)
90
91                  mask_1 = [1 for _ in range(len(char_list))]
92                  mask_0 = [0 for _ in range(self.maxlen-( len(char_list)))]
93                  attention_mask = mask_1 + mask_0
94                  mask_list.append(attention_mask)
95
96              label_tensor = torch.tensor(label_list)
97              mask_tensor = torch.tensor(mask_list,dtype=torch.uint8)
98              embedding_tensor=torch.tensor(embedding)
99
100             return embedding_tensor,mask_tensor,label_tensor
```

轉成tensor

main.py – 模型建立

```python
#Training Mode
if mode == 'Train':

    #create dir for save model
    assert len(args['save_model_name'].split('.pt')) == 2 ,'模型名稱錯誤:{xxx.pt}'
    dir_name = args['save_model_name'].split('.pt')[0]

    model_path = os.path.join('saved_model',dir_name)
    if not os.path.isdir(model_path):
        os.mkdir(model_path)

    #save parameter
    with open(os.path.join(model_path,'parameter.txt'),'w',encoding='utf-8') as p_file:
        for item in args:
            p_file.write(f"{item}: {args[item]}\n")

    #Load Training data
    print("Loading Training data")
    Train_data_path = args['Train_data_path']
    dataset_train = Dataset(Train_data_path,word2vec,dimesion,mode='Train')
    dataloader_train = DataLoader(dataset_train, collate_fn=dataset_train.collate_fn,
                                  batch_size=batch_size, shuffle=False, drop_last=True)

    print(f"training data size : {dataset_train.__len__()}\n")

    #Load Eval Data
    print("Loading Evaluation data")
    Eval_data_path = args['Eval_data_path']
    dataset_eval = Dataset(Eval_data_path,word2vec,dimesion,mode='Eval')
    dataloader_eval = DataLoader(dataset_eval, collate_fn=dataset_eval.collate_fn,
                                  batch_size=batch_size, shuffle=False, drop_last=False)

    print(f"evaluation data size : {dataset_eval.__len__()} \n")

    #Create Model
    model = BiLSTM_CRF(len(dataset_train.id2tag),lstm_hidden_dim=lstm_hidden_dim,lstm_dropout_rate=lstm_dropout_rate).to(device)
    model.reset_parameters()

    #Create optimizer
    optimizer = Adam(filter(lambda p: p.requires_grad, model.parameters()), lr=lr)
```

模型建立及
優化器設定

package/model.py – BiLSTM+CRF

```python
# -*- coding: utf-8 -*-
from torch import nn
from package.nn import ConditionalRandomField
import os
import torch


class BiLSTM_CRF(nn.Module):
    """BiLSTM + CRF
    """

    def __init__(self, tag_set_size, lstm_hidden_dim=256, lstm_dropout_rate=0.1):
        super(BiLSTM_CRF, self).__init__()

        self.bilstm = nn.LSTM(300, lstm_hidden_dim // 2,
                              num_layers=2, bidirectional=True, dropout=lstm_dropout_rate, batch_first=True)
        self.hidden2tag = nn.Linear(lstm_hidden_dim, tag_set_size)
        self.crf = ConditionalRandomField(tag_set_size)

        self.lstm_hidden_dim = lstm_hidden_dim

    def reset_parameters(self):
        self.crf.reset_parameters()

    def loss(self, input: torch.LongTensor, mask: torch.ByteTensor):

        x = input.float()
        x, _ = self.bilstm(x)
        x = self.hidden2tag(x)

        return self.crf(x, mask)

    def forward(self, input: torch.LongTensor, mask: torch.ByteTensor, target: torch.LongTensor):
        x = input.float()
        x, _ = self.bilstm(x)
        x = self.hidden2tag(x)
        return self.crf.neg_log_likelihood_loss(x,mask, target)
```

main.py-訓練模型

```python
for epoch in range(EPOCH):
    print(f"\nEpoch: {epoch}==========================================================================================")

    #Training
    model.train()
    print("---------------------------Training---------------------------")
    with tqdm(desc='Train', total=len(dataloader_train)) as t:
        for i, (input,mask,label) in enumerate(dataloader_train):
            input,mask, label = [_.to(device) for _ in (input,mask,label)]
            loss = model(input,mask,label)
            loss.backward()

            #更新tqdm資訊
            t.update(1)
            t.set_postfix(loss=float(loss))

            # 梯度裁剪，避免出現梯度爆炸情況
            torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)

            optimizer.step()
            # scheduler.step()
            optimizer.zero_grad()
```

訓練模型

main.py-訓練模型

```python
for epoch in range(EPOCH):
    print(f"\nEpoch: {epoch}==========================================================================================")

    #Training
    model.train()
    print("---------------------------Training---------------------------")
    with tqdm(desc='Train', total=len(dataloader_train)) as t:
        for i, (input,mask,label) in enumerate(dataloader_train):
            input,mask, label = [_.to(device) for _ in (input,mask,label)]
            loss = model(input,mask,label)
            loss.backward()

            #更新tqdm資訊
            t.update(1)
            t.set_postfix(loss=float(loss))

            # 梯度裁剪，避免出現梯度爆炸情況
            torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)

            optimizer.step()
            # scheduler.step()
            optimizer.zero_grad()
```

更新模型參數

main.py - 驗證模型

```
140         #Eval
141         model.eval()
142         print('-------------------------Evaluation-------------------------')
143         pred_list = list()
144         true_list = list()
145         with torch.no_grad():
146 
147             for i, (input,mask,label) in enumerate(tqdm(dataloader_eval, desc='Eval')):
148                 #丟入模型預測
149                 input, mask, label = [_.to(device) for _ in (input,mask,label)]
150                 y_pred = model.loss(input,mask)
151 
152                 #抓模型預測label
153                 y_pred = decode_tags_from_ids(y_pred,dataset_eval.id2tag)
154                 for sent in y_pred:
155                     for tag in sent:
156                         pred_list.append(tag)
157 
158 
159                 y_true = list()
160                 for s in range(len(y_pred)):
161                     sentence_list = list()
162                     for p,l in zip(y_pred[s],label[s]):
163                         sentence_list.append(l)
164                     y_true.append(sentence_list)
165 
166                 #抓答案
167                 y_true = decode_tags_from_ids(y_true,dataset_eval.id2tag)
168                 for sent in y_true:
169                     for tag in sent:
170                         true_list.append(tag)
171 
172             #印出評分
173             prec, rec, f1 = evaluate(true_list,pred_list,file_name=model_path)
174             print(f"f1: {f1}")
175             print(f"recall: {rec}")
176             print(f"precision: {prec}")
177 
178         #儲存model
179         model_arg_file_name = args['save_model_name']
180         save_model_name = os.path.join(model_path,model_arg_file_name)
181         torch.save(model.state_dict(), save_model_name)
182         print('Training Finish')
```

預測資料

main.py - 驗證模型

```
        #Eval
        model.eval()
        print('-------------------------Evaluation-------------------------')
        pred_list = list()
        true_list = list()
        with torch.no_grad():

            for i, (input,mask,label) in enumerate(tqdm(dataloader_eval, desc='Eval')):
                #丟入模型預測
                input, mask, label = [_.to(device) for _ in (input,mask,label)]
                y_pred = model.loss(input,mask)

                #抓模型預測label
                y_pred = decode_tags_from_ids(y_pred,dataset_eval.id2tag)
                for sent in y_pred:
                    for tag in sent:
                        pred_list.append(tag)


                y_true = list()
                for s in range(len(y_pred)):
                    sentence_list = list()
                    for p,l in zip(y_pred[s],label[s]):
                        sentence_list.append(l)
                    y_true.append(sentence_list)

                #抓答案
                y_true = decode_tags_from_ids(y_true,dataset_eval.id2tag)
                for sent in y_true:
                    for tag in sent:
                        true_list.append(tag)

                #印出評分
                prec, rec, f1 = evaluate(true_list,pred_list,file_name=model_path)
                print(f"f1: {f1}")
                print(f"recall: {rec}")
                print(f"precision: {prec}")

    #儲存model
    model_arg_file_name = args['save_model_name']
    save_model_name = os.path.join(model_path,model_arg_file_name)
    torch.save(model.state_dict(), save_model_name)
    print('Training Finish')
```

顯示驗證集分數

main.py – 儲存模型

```
140         #Eval
141         model.eval()
142         print('-------------------------Evaluation-------------------------')
143         pred_list = list()
144         true_list = list()
145         with torch.no_grad():
146 
147             for i, (input,mask,label) in enumerate(tqdm(dataloader_eval, desc='Eval')):
148                 #丟入模型預測
149                 input, mask, label = [_.to(device) for _ in (input,mask,label)]
150                 y_pred = model.loss(input,mask)
151 
152                 #抓模型預測label
153                 y_pred = decode_tags_from_ids(y_pred,dataset_eval.id2tag)
154                 for sent in y_pred:
155                     for tag in sent:
156                         pred_list.append(tag)
157 
158                 #將pad的label消除
159                 y_true = list()
160                 for s in range(len(y_pred)):S
161                     sentence_list = list()
162                     for p,l in zip(y_pred[s],label[s]):
163                         sentence_list.append(l)
164                     y_true.append(sentence_list)
165 
166                 #抓答案
167                 y_true = decode_tags_from_ids(y_true,dataset_eval.id2tag)
168                 for sent in y_true:
169                     for tag in sent:
170                         true_list.append(tag)
171 
172             #印出評分
173             prec, rec, f1 = evaluate(true_list,pred_list,file_name=model_path)
174             print(f"f1: {f1}")
175             print(f"recall: {rec}")
176             print(f"precision: {prec}")
177 
178         #儲存model
179         model_arg_file_name = args['save_model_name']
180         save_model_name = os.path.join(model_path,model_arg_file_name)
181         torch.save(model.state_dict(), save_model_name)
182         print('Training Finish')
```

儲存訓練
完成模型

Steps 5. BiLSTM + CRF 測試

BiLSTM+CRF

python main.py --mode Test

讀取訓練好的模型及測試資料

```
184  #Predict Mode
185  elif mode == 'Test':
186
187      #load model name
188      load_model_path = os.path.join('saved_model',args['load_model_name'],args['load_model_name']+'.pt')
189
190      #pred txt name
191      Pred_save_path = os.path.join( 'Predict' ,args['predict_name'])
192
193      #Load Test Data
194      print("Loading Testing data")
195      Test_data_path = args['Test_data_path']
196      dataset_test = Dataset(Test_data_path,word2vec,dimesion,mode='Test')
197      dataloader_test = DataLoader(dataset_test, collate_fn=dataset_test.collate_fn,
198                                   batch_size=batch_size, shuffle=False, drop_last=False)
199
200      print(f"testing data size : {dataset_test.__len__()} \n")
201
202      #Load PreTrain Model
203      model = BiLSTM_CRF(len(dataset_test.id2tag),lstm_hidden_dim=lstm_hidden_dim, lstm_dropout_rate=lstm_dropout_rate).to(device)
204      model.load_state_dict(torch.load(load_model_path))
205      #Test
206      model.eval()
207      print('-------------------------Testing-------------------------')
208
209      with torch.no_grad():
210          sum_pred_list = list()
211
212          for i, (input, mask) in enumerate(tqdm(dataloader_test, desc='Test')):
213              input, mask = [_.to(device) for _ in (input, mask)]
214              y_pred = model.loss(input, mask)
215              y_pred = decode_tags_from_ids(y_pred,dataset_test.id2tag)
216              sum_pred_list += y_pred
217
218
219          with open(Pred_save_path,'w',encoding='utf-8') as file:
220              for char_list,pred_list in zip(dataset_test.data['sentence'],sum_pred_list):
221                  for char,pred in zip(char_list,pred_list):
222                      row = f"{char} {pred}\n"
223                      file.write(row)
224                  file.write('\n')
```

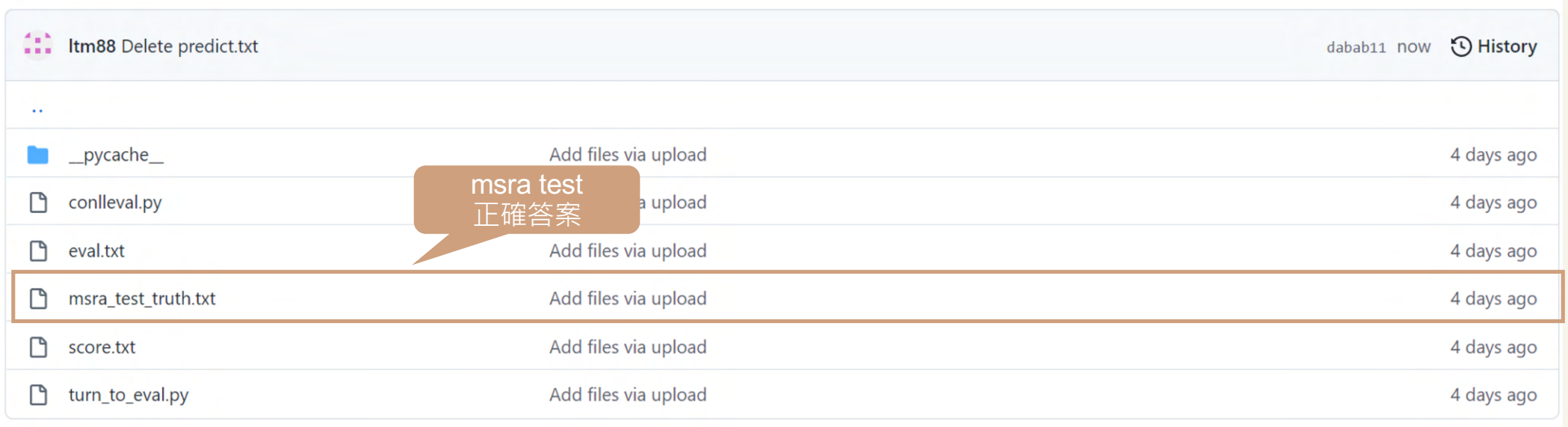
BiLSTM+CRF

```python
#Predict Mode
elif mode == 'Test':

    #load model name
    load_model_path = os.path.join('saved_model',args['load_model_name'],args['load_model_name']+'.pt')

    #pred txt name
    Pred_save_path = os.path.join( 'Predict' ,args['predict_name'])

    #Load Test Data
    print("Loading Testing data")
    Test_data_path = args['Test_data_path']
    dataset_test = Dataset(Test_data_path,word2vec,dimesion,mode='Test')
    dataloader_test = DataLoader(dataset_test, collate_fn=dataset_test.collate_fn,
                                 batch_size=batch_size, shuffle=False, drop_last=False)

    print(f"testing data size : {dataset_test.__len__()} \n")

    #Load PreTrain Model
    model = BiLSTM_CRF(len(dataset_test.id2tag),lstm_hidden_dim=lstm_hidden_dim, lstm_dropout_rate=lstm_dropout_rate).to(device)
    model.load_state_dict(torch.load(load_model_path))
    #Test
    model.eval()
    print('-------------------------Testing-------------------------')

    with torch.no_grad():
        sum_pred_list = list()

        for i, (input, mask) in enumerate(tqdm(dataloader_test, desc='Test')):
            input, mask = [_.to(device) for _ in (input, mask)]
            y_pred = model.loss(input, mask)
            y_pred = decode_tags_from_ids(y_pred,dataset_test.id2tag)
            sum_pred_list += y_pred


        with open(Pred_save_path,'w',encoding='utf-8') as file:
            for char_list,pred_list in zip(dataset_test.data['sentence'],sum_pred_list):
                for char,pred in zip(char_list,pred_list):
                    row = f"{char} {pred}\n"
                    file.write(row)
                file.write('\n')
```

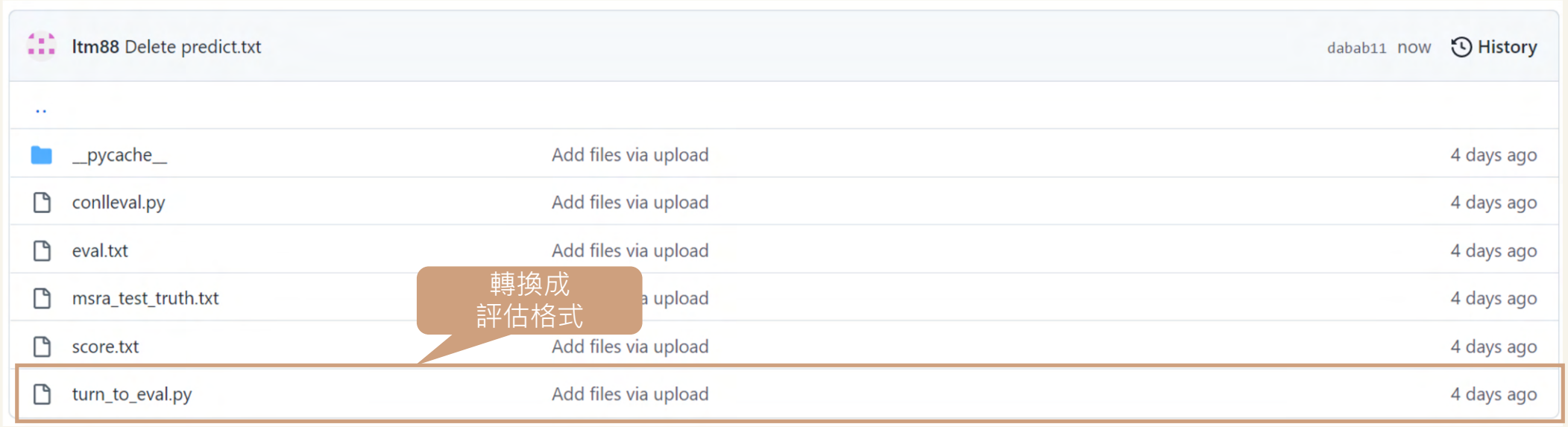
預測資料並儲存結果

Steps 6. 模型效能評估

eval資料夾介紹


ltm88 Delete predict.txt
dabab11 now
History
..
__pycache__ Add files via upload 4 days ago
conlleval.py Add files via upload 4 days ago
eval.txt Add files via upload 4 days ago
msra_test_truth.txt Add files via upload 4 days ago
score.txt Add files via upload 4 days ago
turn_to_eval.py Add files via upload 4 days ago
msra test
正確答案

eval資料夾介紹


ltm88 Delete predict.txt
dabab11 now
History
..
__pycache__ Add files via upload 4 days ago
conlleval.py Add files via upload 4 days ago
eval.txt Add files via upload 4 days ago
msra_test_truth.txt Add files via upload 4 days ago
score.txt Add files via upload 4 days ago
turn_to_eval.py Add files via upload 4 days ago
轉換成
評估格式

eval.txt

msra_test_truth.txt

```
王 B-PER
羲 I-PER
之 I-PER
的 O
《 O
兰 O
亭 O
序 O
》 O
、 O
颜 B-PER
筋 O
柳 B-PER
骨 O
、 O
颠 O
张 B-PER
醉 O
素 B-PER
```

predict.txt

```
王 B-PER
羲 I-PER
之 I-PER
的 O
《 O
兰 B-LOC
亭 O
序 O
》 O
、 O
颜 O
筋 O
柳 O
骨 O
、 O
颠 O
张 O
醉 O
素 O
```

正確答案

預測字元

預測答案

```
王 B-PER B-PER
羲 I-PER I-PER
之 I-PER I-PER
的 O O
《 O O
兰 O B-LOC
亭 O O
序 O O
》 O O
、 O O
颜 B-PER O
筋 O O
柳 B-PER O
骨 O O
、 O O
颠 O O
张 B-PER O
醉 O O
素 B-PER O
```

turn_to_eval.py

python turn_to_eval.py --truth truth.txt --prediction ../Predict/predict.txt

```python
#-*- coding=utf-8 -*-
import argparse
import pandas as pd
import codecs
import sys
from conlleval import *
sys.stdout = codecs.getwriter("utf-8")(sys.stdout.detach())
parser = argparse.ArgumentParser()
parser.add_argument("--truth",default='msra_test_truth.txt',type=str,help="your truth")
parser.add_argument("--prediction",default='../Predict/predict.txt',type=str,help='your prediction')
args = parser.parse_args()
args = vars(args)
#read file
df_truth =  pd.read_csv(args["truth"],encoding='utf-8',delimiter=' ',header=None,skip_blank_lines =False)
df_prediction =  pd.read_csv(args["prediction"],encoding='utf-8',delimiter=' ',header=None,skip_blank_lines =False)
df = pd.concat([df_truth,df_prediction[1]],axis=1)
df.to_csv("eval.txt",encoding='utf-8',header=None,index=None,sep=" ")

```

score.txt

python conlleval.py < eval.txt

```
processed 243181 tokens with 6749 phrases; found: 6360 phrases; correct: 4320.
accuracy:  73.67%; (non-O)
accuracy:  96.48%; precision:  67.92%; recall:  64.01%; FB1:  65.91
              LOC: precision:  74.27%; recall:  63.05%; FB1:  68.20  2771
              ORG: precision:  52.04%; recall:  59.42%; FB1:  55.49  1866
              PER: precision:  74.93%; recall:  69.75%; FB1:  72.24  1723
```

https://github.com/sighsmile/conlleval

Q & A 綜合討論

The END

http://nlp.ee.ncu.edu.tw

Source: wikimedia commons