AUTONLU: An On-demand Cloud-based Natural Language Understanding System for Enterprises

Nham Le [1,3] * **Tuan Manh Lai** [2,3] * **Trung Bui** [3] **Doo Soon Kim** [3]

[1] University of Waterloo, Ontario, Canada
[2] University of Illinois at Urbana-Champaign, USA
[3] Adobe Research, San Jose, USA

Abstract

With the renaissance of deep learning, neural networks have achieved promising results on many natural language understanding (NLU) tasks. Even though the source codes of many neural network models are publicly available, there is still a large gap from open-sourced models to solving real-world problems in enterprises. Therefore, to fill this gap, we introduce AUTONLU, an on-demand cloud-based system with an easy-to-use interface that covers all common use-cases and steps in developing an NLU model. AUTONLU has supported many product teams within Adobe with different use-cases and datasets, quickly delivering them working models. To demonstrate the effectiveness of AUTONLU, we present two case studies. i) We build a practical NLU model for handling various image-editing requests in Photoshop. ii) We build powerful keyphrase extraction models that achieve state-of-the-art results on two public benchmarks. In both cases, end users only need to write a small amount of code to convert their datasets into a common format used by AUTONLU.

1 Introduction

In recent years, many deep learning methods have achieved impressive results on a wide range of tasks, ranging from question answering (Seo et al., 2017; Lai et al., 2018b) to named entity recognition (NER) (Lin et al., 2019; Jiang et al., 2019) to intent detection and slot filling (Wang et al., 2018; Chen et al., 2019). Even though the source codes of many models are publicly available, going from an open-sourced implementation of a model for a public dataset to a production-ready model for an in-house dataset is not a simple task. Furthermore, in an enterprise, only few engineers are familiar with deep learning research and frameworks. Therefore, to facilitate the development and adoption of deep learning models within Adobe, we introduce a new system named AUTONLU. It is an on-demand cloud-based system that enables multiple users to create and edit datasets and to train and test different state-of-the-art NLU models. AUTONLU's main principles are:

- **Ease of use**. AUTONLU aims to help users with limited technical knowledge to train and test models on their datasets. We provide GUI modules to accommodate the most common use-cases, from creating/cleaning a dataset to training/evaluating/debugging a model.
- **State-of-the-art models**. Users should not sacrifice performance for ease-of-use. Our built-in models provide state-of-the-art performance on multiple public datasets. AUTONLU also supports hyperparameter tuning using grid search, allowing users to fine-tune the models even further.
- **Scalability**. AUTONLU aims to be deployed in enterprises where computing costs could be a limiting factor. We provide an on-demand architecture so that the system could be utilized as much as possible.

At Adobe, AUTONLU has been used to train NLU models for different product teams, ranging from Photoshop to Document Cloud. To demonstrate the effectiveness of AUTONLU, we present two case studies. i) We build a practical NLU model for handling various image-editing requests in Photoshop. ii) We build powerful keyphrase extraction models that achieve state-of-the-art results on two public benchmarks. In both cases, end users only need to write a small amount of code to convert their datasets into a common format used by AUTONLU.

[*] Equal contributions. The work was conducted while the first two authors interned at Adobe Research.

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations, pages 8–13
December 4 - 7, 2020. ©2020 Association for Computational Linguistics

2 Related work

Closely related branches of work to ours are toolkits and frameworks designed to provide a suite of state-of-the-art NLP models to users (Gong et al., 2019; Akbik et al., 2019; Wang et al., 2019; Zhu et al., 2020; Qi et al., 2020). However, several of these works do not have a user-friendly interface. For example, `Flair` (Akbik et al., 2019), `NeuronBlocks` (Gong et al., 2019), and `jiant` (Wang et al., 2019) require users to work with command-line interfaces. Different from these works, an end-user with no programming skill can still create powerful NLU models using our system. Furthermore, most previous works are not explicitly designed for enterprise settings where use-cases and business needs can be vastly different from team to team. On the other hand, since AUTONLU is an on-demand cloud-based system, it provides more flexibility to end users.

In 2018, Google introduced AutoML Natural Language[1], a platform that enables users to build and deploy machine learning models for various NLP tasks. Our system is different from AutoML in the following aspects. First, AutoML uses neural architecture search (NAS) (Elsken et al., 2019) to find the best model for the task of interest. As users are not allowed to simply choose an existing architecture, the process can be time-consuming even for simple tasks (e.g., 2∼3 hours). On the other hand, AUTONLU provides a rich gallery of existing architectures for NLU. In future work, we are also planning to integrate NAS into AUTONLU. Second, as a self-hosted solution, AUTONLU provides product teams of Adobe with total control over their datasets and trained models. This enhances privacy and provides more flexibility at the same time. For example, as of writing, there is no way to download a trained model from AutoML to a local machine to use it for a subsequent task. AUTONLU supports it out-of-the-box.

3 AUTONLU

3.1 Components and architecture

Figure 1 shows the overall architecture of our system. There are 3 main components:

- **A web application** that serves as the frontend to the users. The most important component of the application is a Scheduler that monitors the status of the cluster, then assigns jobs to the most appropriate instances, as well as spawns more/shuts off instances based on the workload to minimize the computing costs. The user interface is discussed in more detail in Section 3.3.

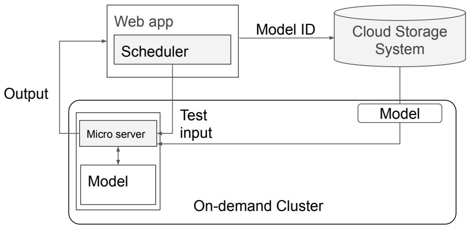


Figure 1: AUTONLU architecture. In the figure is the dataflow when the user calls to the `/test` endpoint.

- **A cloud storage system** that stores datasets, large pre-trained language models (e.g., BERT (Devlin et al., 2018)), trained NLU models, and models' metadata. We use Amazon S3 as our storage system, due to its versioning support and data transfer speed to EC2 instances.
- **An on-demand cluster** that performs the actual training and testing. While the Lambda computing model seems to be a better fit at first thought, after careful consideration, we choose EC2 instances to prioritize user experience over some costs: in our setting, we have multiple concurrent users with small to medium datasets. If the training itself takes only 10 minutes, any amount of wait time is significant. By maintaining a certain number of always-on instances, users will always have instant interaction with the system without any delay. Cluster's instances are initiated using prebuilt images, which we discuss in Section 3.2.

3.2 Instance image

Regardless of the underlying model, in each prebuilt image, an included webserver is configured to serve the following endpoints:

- `/train` that connects to the training code of the underlying model.
- `/is_free` that returns various information about the utilization of the instance (e.g, GPU memory usage).
- `/test` that connects to the testing code of the underlying model.
- `/notebook` that connects to the Jupyter Lab notebook's URL packaged in the image.

[1] `https://cloud.google.com/natural-language`

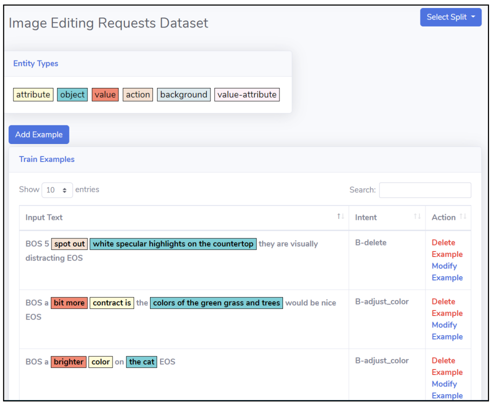


Image Editing Requests Dataset

Entity Types

attribute object value action background value-attribute

Train Examples

Show 10 entries

Search:

Input Text	Intent	Action
BOS 5 spot out white specular highlights on the countertop they are visually distracting EOS	B-delete	Delete Example Modify Example
BOS a bit more contract is the colors of the green grass and trees would be nice EOS	B-adjust_color	Delete Example Modify Example
BOS a brighter color on the cat EOS	B-adjust_color	Delete Example Modify Example

Figure 2: Dataset view of AUTONLU.

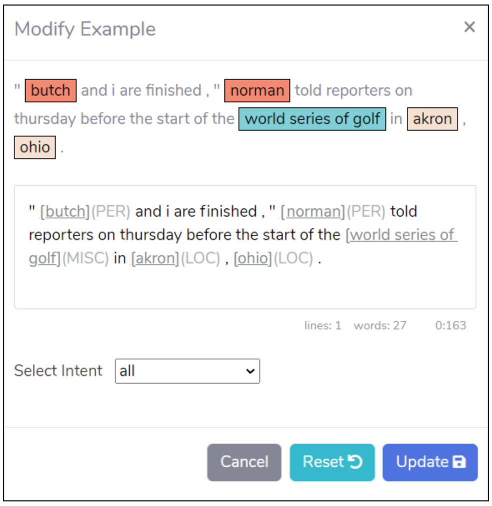
Each image also exposes an SSH connection, authenticated using LDAP. Experienced users can also make use of the packaged TensorBoard to monitor the training process.

3.3 User Interface

3.3.1 Dataset Tool

Public and internal datasets come in many different formats, as they may have been collected for many years and annotated in different ways. To mitigate that, we develop an intermediate representation (IR) that is suitable for many NLU tasks and write frontends to convert common dataset formats to said IR. We also provide a converter that converts this IR back into other dataset formats, making converting a dataset from one format to another trivial. In our setting (an enterprise environment), a dataset frontend converter is the only part that may need to be written by an end-user, and we believe that it is significantly simpler than building the whole NLU pipeline.

Figure 2 shows the dataset view. Visualizing and editing datapoints are straightforward, and do not depend on the source/target dataset format (Figure 3). While it is not common to edit a public dataset, the same is typically not true for internal datasets. Internal datasets may need to be modified and expanded based on business needs and use-cases.



Modify Example

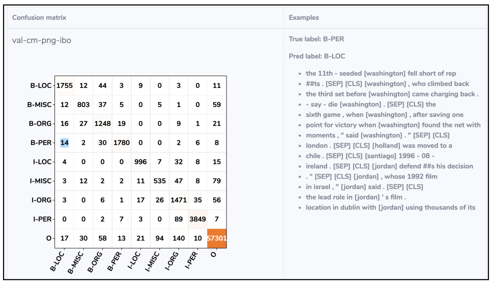
" butch and i are finished , " norman told reporters on thursday before the start of the world series of golf in akron , ohio .

" [butch](PER) and i are finished , " [norman](PER) told reporters on thursday before the start of the [world series of golf](MISC) in [akron](LOC) , [ohio](LOC) .

lines: 1 words: 27 0:163

Select Intent all

Figure 3: Edit/Add a datapoint.



Confusion matrix

val-cm-png-ibo

	B-LOC	B-MISC	B-ORG	B-PER	I-LOC	I-MISC	I-ORG	I-PER	O
B-LOC	1755	12	44	3	9	0	3	0	11
B-MISC	12	803	37	5	0	5	1	0	59
B-ORG	16	27	1248	19	0	0	9	1	21
B-PER	14	2	30	1780	0	0	2	6	8
I-LOC	4	0	0	0	996	7	32	8	15
I-MISC	3	12	2	2	11	535	47	8	79
I-ORG	3	0	6	1	17	26	1471	35	56
I-PER	0	0	2	7	3	0	89	3849	7
O	17	30	58	13	21	94	140	10	57301

Examples

True label: B-PER

Pred label: B-LOC

- the 11th - seeded [washington] fell short of rep
- ##ts . [SEP] [CLS] [washington] , who climbed back
- the third set before [washington] came charging back .
- - say - die [washington] . [SEP] [CLS] the
- sixth game , when [washington] , after saving one
- point for victory when [washington] found the net with
- moments , " said [washington] . " [SEP] [CLS]
- london . [SEP] [CLS] [holland] was moved to a
- chile . [SEP] [CLS] [santiago] 1996 - 08 -
- ireland . [SEP] [CLS] [jordan] defend ##s his decision
- . " [SEP] [CLS] [jordan] , whose 1992 film
- in israel , " [jordan] said . [SEP] [CLS]
- the lead role in [jordan] ' s film .
- location in dublin with [jordan] using thousands of its

Figure 4: An example interactive confusion matrix.

3.3.2 Analysis Tool

We include TensorBoard in our prebuilt images to display common training metrics. However, since our main users are typically product teams with limited experience in machine learning, we also develop interactive views to analyze the trained results. For example, Figure 4 shows our interactive confusion matrix view: rather than just knowing that there are 14 instances in which a mention with the label “Person” is misclassified as “Location”, users can click on a cell in the matrix to see which instances are misclassified. This is even more important for internal datasets: the errors may actually be in the dataset instead of the model, and we can catch it using this view. In fact, as we will demonstrate in Section 4.1, we have caught many labeling errors in our internal datasets using this tool.

3.3.3 Resource Management Tool

In most use-cases, AUTONLU automatically handles resource management for the users. However, if an advanced user wants to manually manage instances' life cycle, assign a task to a specific instance, or to debug an instance, we provide a GUI to do so as well. Concretely, we provide the following functionalities:

- *Create an instance with a desired hardware configuration and docker image*. By default, AUTONLU creates an instance with 4 CPU cores, 8 GBs of RAM, and 1 NVIDIA V100 GPU, which are all configurable to the user's desire. The default docker image is the one containing all the supported models, but users can choose from one of the prebuilt images that contains just a single model if that's their use-case.

- *Assign a task to an instance.* During training and testing, users can choose whether to let AutoNLU to distribute the task or to assign the task to a specific instance: it is common for a product team to reserve a few instances for themselves and want to use just those instances.
- *Access an instance's shell and files.* Since Ease-of-use is one of our core design principles, we package in all of our prebuilt images a Jupyter Lab server, with the intention of using it as a lightweight IDE/shell environment. While we also expose SSH connection to each instance, we expect users to find the Jupyter Lab a more friendly approach.

4 Case studies

4.1 NLU Models for Image-Editing Requests

One of the first clients of AutoNLU was the Photoshop team, as we want to build a chatbot using their image-editing requests dataset (Manuvinakurike et al., 2018; Brixey et al., 2018). The dataset was collected in many years, annotated both using Amazon Mechanical Turk and by our in-house annotators. Cleaning this dataset is a challenge in itself, and in this case study, we aim to create an effective workflow to train a state-of-the-art model and clean the dataset at the same time.

We first convert the dataset into our IR, and train a simple model using the fastest algorithm provided by AutoNLU. This initial model provides us with a rough confusion matrix, and we manually inspect cells with the biggest values. Those cells give us an insight into some systematic labeling errors, such as in Figure 5. We then fix those labeling errors, either by using the dataset interface in AutoNLU , or by writing scripts. With this new dataset, we retrain another model and repeat the process.

Once the fast model performance is comparable to its performance on some public datasets, such as ATIS (Hemphill et al., 1990), we switch to train and fine-tune a bigger model. More specifically, we employ a joint intent classification and slot filling model based on BERT (Chen et al., 2019), which is already implemented in AutoNLU. By the end of this process, we end up with a powerful NLU model, as reported in Table 1, and a cleaned dataset that is useful for subsequent tasks. The NLU model created using AutoNLU outperforms a competing model created using RASA (Bocklisch et al., 2017) and a joint model of intent determination and slot filling (JIS) (Zhang and Wang, 2016) by a large margin.

```
True label: B-adjust_brightness
Pred label: B-adjust_color
[[CLS] light ##en the vegetables [SEP]]
[[CLS] make the dirt darker in brown
    color [SEP]]
```

Figure 5: 2 labeling errors captured by the interactive confusion matrix near the end of the training-cleaning process. The ## is the artifact from BERT tokenizer.

Model	Metrics			
	Intent	SP	SR	SF1
JIS (2016)	0.832	0.850	0.726	0.783
RASA	0.924	0.833	0.605	0.701
AutoNLU	**0.954**	**0.869**	**0.854**	**0.862**

Table 1: Results on the image-editing requests dataset. Intent accuracy, slot precision, slot recall, and slot F1 scores are reported. Scores of our models are averaged over three random seeds.

4.2 Keyphrase Extraction Models

Keyphrase extraction is the task of automatically extracting a small set of phrases that best describe a document. As keyphrases provide a high-level summarization of the considered document and they give the reader some clues about its contents, keyphrase extraction is a problem of great interest to the Document Cloud team of Adobe. In this case study, we aim to develop an effective keyphrase extraction system for the team.

Similar to recent works on keyphrase extraction (Sahrawat et al., 2020), we formulate the task as a sequence labeling task. Given an input sequence of tokens $\mathbf{x} = \{x_1, x_2, ..., x_n\}$, the goal is to predict a sequence of labels $\mathbf{y} = \{y_1, y_2, ..., y_n\}$ where $y_i \in \{\mathtt{B}, \mathtt{I}, \mathtt{O}\}$. Here, label B denotes the beginning of a keyphrase, I denotes the continuation of a keyphrase, and O corresponds to tokens that are not part of any keyphrase. This formulation is naturally supported by our platform, as the task of slot filling in NLU is basically a sequence labeling task. We first collect two public datasets for keyphrase extraction: Inspec (Hulth, 2003) and SE-2017 (Augenstein et al., 2017). We then convert them to the common intermediate representation. After that, we simply use AutoNLU to train and tune models. We employ the BiLSTM-CRF archi-

Model	Datasets	
	Inspec	SE-2017
KEA (2005)	0.137	0.129
TextRank (2004)	0.122	0.157
SingeRank (2008)	0.123	0.155
SGRank (2015)	0.271	0.211
Transformer (2020)	0.595	0.522
BERT (AUTONLU)	0.596	0.537
SciBERT (AUTONLU)	**0.598**	**0.544**

Table 2: Results on Inspec and SE-2017 datasets. F1 scores are reported. Scores of our models are averaged over three random seeds.

tecture (Huang et al., 2015) that is already available in AUTONLU. We experiment with two different pre-trained language models as the first embedding layer: BERT (Devlin et al., 2018) and SciBERT (Beltagy et al., 2019). Table 2 shows the results on the datasets. We see that both models created using AUTONLU outperform previous models for the task, achieving new state-of-the-art results. As AUTONLU can automatically perform hyperparameter tuning using grid search, models produced by AUTONLU typically have satisfying performance (assuming that the selected underlying architecture is expressive enough). It is worth noting that during this entire process, the only code we need to write is for converting the Inspec and SE-2017 datasets to the IR.

5 Conclusion

In this work, we introduce AUTONLU, an on-demand cloud-based platform that is easy-to-use and has enabled many product teams within Adobe to create powerful NLU models. Our design principles make it an ideal candidate for enterprises who want to have an NLU system for themselves, with minimal deep learning expertise. AUTONLU 's code is in the process to be open-sourced, and we invite contributors to contribute. In future work, we will implement more advanced features such as transfer learning, knowledge distillation and neural architecture search, which have been shown to be useful in building real-world NLP systems (Lai et al., 2018a; Jiang et al., 2019; Lai et al., 2019, 2020; Klyuchnikov et al., 2020). Furthermore, we will extend our system to have more advanced analytics features (Murugesan et al., 2019), and to better support other languages (Nguyen and Nguyen, 2020).

References

A. Akbik, T. Bergmann, Duncan Blythe, K. Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL-HLT*.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: Scienceie - extracting keyphrases and relations from scientific publications. *CoRR*, abs/1704.02853.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *EMNLP/IJCNLP*.

Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open source language understanding and dialogue management. *ArXiv*, abs/1712.05181.

Jacqueline Brixey, Ramesh Manuvinakurike, Nham Le, Tuan Lai, Walter Chang, and Trung Bui. 2018. A system for automated image editing from natural language commands. *arXiv preprint arXiv:1812.01083*.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *ArXiv*, abs/1902.10909.

Soheil Danesh, Tamara Sumner, and James H. Martin. 2015. Sgrank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction. In **SEM@NAACL-HLT*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *ArXiv*, abs/1808.05377.

Ming Gong, Linjun Shou, Wutao Lin, Zhijie Sang, Quanjia Yan, Ze Yang, and Daxin Jiang. 2019. Neuronblocks - building your nlp dnn models like playing lego. *ArXiv*, abs/1904.09535.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *EMNLP*.

Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. 2019. Improved differentiable architecture search for language modeling and named entity recognition. In *EMNLP/ICJNLP*.

Nikita Klyuchnikov, Ilya Trofimov, Ekaterina Artemova, Mikhail Salnikov, Maxim Fedorov, and Evgeny Burnaev. 2020. Nas-bench-nlp: Neural architecture search benchmark for natural language processing. *arXiv preprint arXiv:2006.07116*.

Tuan Lai, Trung Bui, Nedim Lipka, and Sheng Li. 2018a. Supervised transfer learning for product information question answering. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1109–1114. IEEE.

Tuan Lai, Quan Hung Tran, Trung Bui, and Daisuke Kihara. 2019. A gated self-attention memory network for answer selection. *arXiv preprint arXiv:1909.09696*.

Tuan Manh Lai, Trung Bui, and Sheng Li. 2018b. A review on deep learning techniques applied to answer selection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2132–2144, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Tuan Manh Lai, Quan Hung Tran, Trung Bui, and Daisuke Kihara. 2020. A simple but effective bert model for dialog state tracking on resource-limited systems. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8034–8038. IEEE.

Ying Lin, Liyuan Liu, Heng Ji, Dong Yu, and Jiawei Han. 2019. Reliability-aware dynamic feature composition for name tagging. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 165–174, Florence, Italy. Association for Computational Linguistics.

Ramesh R. Manuvinakurike, Jacqueline Brixey, Trung Bui, W. Chang, Doo Soon Kim, Ron Artstein, and Kallirroi Georgila. 2018. Edit me: A corpus and a framework for understanding natural language image editing. In *LREC*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *EMNLP*.

Sugeerth Murugesan, Sana Malik, Fan Du, Eunyee Koh, and Tuan Manh Lai. 2019. Deepcompare: Visual and interactive comparison of deep learning model performance. *IEEE computer graphics and applications*, 39(5):47–59.

Dat Quoc Nguyen and Anh Tuan Nguyen. 2020. Phobert: Pre-trained language models for vietnamese. *arXiv preprint arXiv:2003.00744*.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *ACL*.

Dhruva Sahrawat, Debanjan Mahata, Haimin Zhang, Mayank Kulkarni, Agniv Sharma, Rakesh Gosangi, Amanda Stent, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. 2020. Keyphrase extraction as sequence labeling using contextualized embeddings. In *European Conference on Information Retrieval*, pages 328–335. Springer.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ArXiv*, abs/1611.01603.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*.

Alex Wang, Ian F. Tenney, Yada Pruksachatkun, Katherin Yu, Jan Hula, Patrick Xia, Raghu Pappagari, Shuning Jin, R. Thomas McCoy, Roma Patel, Yinghui Huang, Jason Phang, Edouard Grave, Haokun Liu, Najoung Kim, Phu Mon Htut, Thibault F'evry, Berlin Chen, Nikita Nangia, Anhad Mohananey, Katharina Kann, Shikha Bordia, Nicolas Patry, David Benton, Ellie Pavlick, and Samuel R. Bowman. 2019. `jiant` 1.2: A software toolkit for research on general-purpose text understanding models. `http://jiant.info/`.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based rnn semantic frame parsing model for intent detection and slot filling. *ArXiv*, abs/1812.10235.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 2005. Kea: Practical automated keyphrase extraction. In *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*, pages 129–152. IGI global.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI*.

Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jin chao Li, Baolin Peng, Jianfeng Gao, Xiao-Yan Zhu, and Minlie Huang. 2020. Convlab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems. *ArXiv*, abs/2002.04793.