AACL-IJCNLP 2020

The 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics
&
The 10th International Joint Conference on Natural Language Processing

Proceedings of System Demonstrations

December 4 - 7, 2020

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-952148-92-7

Introduction

Welcome to the proceedings of the system demonstrations session. This volume contains the papers of the system demonstrations presented at the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing (AACL-IJCNLP) on December 4 - 7, 2020.

The AACL-IJCNLP 2020 demonstrations track invited submissions ranging from early research prototypes to mature production-ready systems. We received 13 submissions this year, of which 7 were selected for inclusion in the program (acceptance rate of 54%) after review by at least three members of the program committee.

We would like to express our gratitude to the members of the program committee. The candidate papers were selected by the demo chairs based on the feedback received by reviewers.

Demonstrations papers will be presented during the conference in dedicated demo sessions.

Best,

Douwe Kiela
Derek F. Wong
AACL-IJCNLP 2020 Demo Chairs

Demonstration Chairs

Douwe Kiela, Facebook AI Research
Derek Wong, University of Macau

Table of Contents

Conference Program

AMesure: A Web Platform to Assist the Clear Writing of Administrative Texts
Thomas François, Adeline Müller, Eva Rolin and Magali Norré

AutoNLU: An On-demand Cloud-based Natural Language Understanding System for Enterprises
Nham Le, Tuan Lai, Trung Bui and Doo Soon Kim

ISA: An Intelligent Shopping Assistant
Tuan Lai, Trung Bui and Nedim Lipka

metaCAT: A Metadata-based Task-oriented Chatbot Annotation Tool
Ximing Liu, Wei Xue, Qi Su, weiran nie and Wei Peng

NLP Tools for Predictive Maintenance Records in MaintNet
Farhad Akhbardeh, Travis Desell and Marcos Zampieri

Fairseq S2T: Fast Speech-to-Text Modeling with Fairseq
Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko and Juan Pino

NLPStatTest: A Toolkit for Comparing NLP System Performance
Haotian Zhu, Denise Mak, Jesse Gioannini and Fei Xia

AMesure: a web platform to assist the clear writing of administrative texts

Thomas François
IL&C, CENTAL/TeAMM
UCLouvain

Adeline Müller
IL&C, CENTAL
UCLouvain

Eva Rolin
IL&C, CENTAL
UCLouvain

Magali Norré
IL&C, CENTAL
UCLouvain

Abstract

This article presents the AMesure platform, which aims to assist writers of French administrative texts in simplifying their writing. This platform includes a readability formula specialized for administrative texts and it also uses various natural language processing (NLP) tools to analyze texts and highlight a number of linguistic phenomena considered difficult to read. Finally, based on the difficulties identified, it offers pieces of advice coming from official plain language guides to users. This paper describes the different components of the system and reports an evaluation of these components.

1 Introduction

In our current society, written documents play a central role as an information channel, especially in the context of communication between institutions and their target audiences (Madinier, 2009). Unfortunately, although efforts to raise the education level of the population worldwide have increased in recent decades, reports (OECD, 2016) point out that a significant proportion of citizens still have general reading difficulties. As regards administrative texts, various reading issues have been reported. For instance, Kimble (1992) reported that, in a survey carried out in the US, 58% of the respondents admitted to dropping out of an administrative process due to the reading difficulty.

Administrations have been aware of this issue for decades and have launched various initiatives to address it, the most prominent of which is the Plain Language movement. Plain language aims to increase the accessibility of legal documents for a general audience and has been shown to both reduce costs and please readers (Kimble, 1996). It has not only been promoted through various campaigns (e.g. *Plain English Campaign* in the UK) and writing guides (Gouvernement du Québec, 2006; Ministère de la Communauté française de Belgique, 2010; European Union, 2011; Plain Language Action and Information Network, 2011; Cutts, 2020), but also incorporated in some legal principles. However, its widespread application is still undermined due to, for example, the efforts required to train writers (Desbiens, 2008), or the necessity to persuade writers – especially legal ones – to abandon their flowery style, which is seen as a determinant of the image of expertise they project in the reader's mind (Adler, 2012). This second reason, however, falls beyond the scope of the current study, which aims to address the first reason, i.e. writers' training.

Recent research by Nord (2018) revealed that although several plain language guides are available to assist writers of administrative texts in their work, the guidelines provided in these guides are not always followed by writers, mainly because they are too vague and too numerous. To relieve writers from the need to keep all these guidelines in mind, we have designed a web platform, AMesure[1], aimed at automatically identifying clear writing issues in administrative texts and providing simple writing advice that is contextually relevant. In its current state, the platform offers the three following functionalities: (1) providing an overall readability score based on a formula specialized to administrative texts; (2) identifying, in a text, linguistic phenomena that are assumed to have a negative effect on the comprehension of the text; (3) for the phenomena detected in step 2, proposing simplification advice found in plain language guides.

In the following sections, we first refer to some related work (Section 2), before describing the NLP analyses carried out to operate the system (Section 3.1). Then, we introduce the system and the way suggestions are provided (Section 3.2). The paper

[1]The platform is freely available online at https://cental.uclouvain.be/amesure/.

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 1–7

concludes with a report about the system performance (Section 4).

2 Related work

This work stands at the intersection between two very different fields: writing studies – "the interdisciplinary science that studies all the processes and knowledge involved in the production of professional writings and their appropriateness for the addressees" (Labasse, 2001) – and automatic text simplification (ATS), a branch of NLP that aims to automatically adapt difficult linguistic structures while preserving the meaning to enhance text accessibility.

Relevant facts from writing studies have already been covered in the introduction. As regards ATS, the last few years have witnessed the publication of numerous interesting studies, reviewed by Shardlow (2014), Siddharthan (2014), and Saggion (2017). In brief, the field has mainly focused on developing algorithms to automatically simplify complex words (*lexical simplification*) and/or complex syntactic structures (*syntactic simplification*). It has first relied on rule-based approaches (Chandrasekar et al., 1996; Siddharthan, 2011) in which a text is automatically parsed before being applied simplification rules defined by experts. Later, ATS has been assimilated to a translation task (the original version is translated into a simplified version) and addressed with statistical translation systems (Specia, 2010; Zhu et al., 2010). As neural machine translation has emerged under the impulse of deep learning, the Seq2Seq model has become prevalent for ATS too (Nisioi et al., 2017; Zhang and Lapata, 2017).

Some work has specifically focused on the issue of lexical simplification, which involves different techniques. Lexical simplification is generally operated in four steps, the first one being the identification of complex words. Some systems choose to consider all words as candidates for substitution (Bott et al., 2012); others use a list of complex words or machine learning techniques for classification of complex words (Alarcon et al., 2019). Once complex words have been identified, the next step is the generation of simpler synonyms for substitution, either by relying on lexical resources (De Belder and Moens, 2010; Billami et al., 2018), getting candidates from corpora (Coster and Kauchak, 2011), producing them with embeddings (Glavaš and Štajner, 2015; Paetzold and Specia, 2016) or, more recently, with BERT (Qiang et al., 2020). In a next step, the candidates are semantically filtered to fit the context and are ranked according to their difficulty by classifiers using various word characteristics (e.g. frequency, embedding, morphemes, syllabic structures, etc.) (Paetzold and Specia, 2017; Billami et al., 2018; Qiang et al., 2020).

Although numerous ATS systems are described in publications, we have found only four of them that made their way through a web platform tailored to writers' needs. Scarton et al. (2010) developed a simplification web platform for Portuguese, in which the user is able to either accept or reject simplifications done by the system. Similarly, Lee et al. (2016)'s system performs lexical and syntactic simplifications for English and supports human post-editing. More recently, Falkenjack et al. (2017) introduced TeCST, which is able to perform simplification at different levels, depending on the user. Finally, Yimam and Biemann (2018) implemented a semantic writing aid tool able to suggest context-aware lexical paraphrases to writers. None of these tools, however, have focused on writers of administrative texts, nor on French.

AMesure could also be related to the family of writing assistants, such as Word or LibreOffice. However, only a few of them provides writing advice based on specific criteria or plain language guides. There are some examples of these tools available for the general public in French: (1) Plainly[2]; (2) Lirec[3] which relies on the FALC guidelines, an equivalent of the Easy-to-Read language in French, tailored to people with a cognitive disability; or (3) Antidote[4], which offers various writing advice to be clearer and includes five readability indexes. These are however commercial tools, whose scientific foundations are difficult to know and to compare to.

3 The platform

AMesure aims to help writers to produce clear and simple administrative texts for a general audience[5].

For this purpose, it offers various diagnoses about the reading difficulty of a text as well as

[2] https://www.labrador-company.fr/outil-langage-clair/

[3] http://sioux.univ-paris8.fr/lirec/

[4] https://www.antidote.info/fr

[5] People with low reading levels require even more simplified texts (with shorter sentences, no subordinated clauses, etc.). This "oversimplification" falls under the scope of the Easy Language domain.

advice on simpler ways of writing. Before moving to the description of the platform in Section 3.2, we first introduce the various NLP processes used to analyse the text and annotate difficulties in Section 3.1.

3.1 The analysis of the text

As soon as a text is uploaded on the platform, it is processed through various NLP tools to get a rich representation of the text, on which further rule-based processes are then applied. In a first step, the text is split into sentences and POS-tagged with MElt (Denis and Sagot, 2012), before being syntactically parsed with the Berkeley parser adapted for French (Candito et al., 2010). As a result, each sentence is represented as a dependency tree, on which we apply a set of handcrafted rules expressed in the form of regular expressions using the Tregex (Levy and Andrew, 2006) syntax. The rules currently implemented (François et al., 2018) are able to identify four classes of complex syntactic structures: passive clauses, relative clauses, object clauses, and adverbial clauses. Identifying these four classes is motivated by the characteristics of administrative texts. Passive clauses and infinitive verbs are often used in administrative texts to conceal the presence of the writer (Cusin-Berche, 2003), while other types of clause are used to provide the reader with as many detailed information as possible (Catherine, 1968). Parentheticals are also identified, as they are prone to hinder the reading process.

In a second step, the tagged text is further processed to carry out lexical analyses of the text. During this step, three types of lexical difficulties are identified. Firstly, rare words are detected relying on frequencies from Lexique3 (New et al., 2007), based on a threshold set empirically.

Secondly, technical terms are detected with some heuristics able to detect both simple terms and multi-word terms – a task that remains a challenge for current fully automatic approaches (da Silva Conrado et al., 2014) – that are included in a database. The database has been compiled from three different sources: (1) the official lists from the Belgian administration; (2) a list of terms extracted from a corpus of 115 administrative texts following the automatic extraction approach of Chung (2003) and then manually validated; and (3) a book describing various characteristics of the administrative style and listing administrative terms (Catherine, 1968). At the end of the collection phase, we obtained 3,382 terms, some of which could, however, not be considered as difficult (e.g. academy, degree, jury, trainee, etc.). We therefore filtered the resource by excluding words found in the list of the 8000 simplest words in French (Gougenheim et al., 1964). As result, the final term database amounts to 2,481 entries.

Thirdly, abbreviations are automatically detected as they are known to produce reading errors, especially when they are used by specialized writers to communicate to non-specialized readers. For instance, Sinha et al. (2011) report that the Joint Commission on Accreditation of Healthcare Organizations estimated that 5% of medical errors are due to abbreviations. In our system, abbreviations are detected based on an abbreviation database, collected from Belgian public authorities. The database relate the extended version(s) of abbreviations (e.g. *communauté française*, *Institutions publiques de protection de la jeunesse*) with the corresponding abbreviated forms (e.g. *comm. fr.*; *IPPJ* and *I.P.P.J.* respectively). The list provided by public authorities was supplemented via a semi-automatic extraction process applied to our corpus of 115 administrative texts. This extraction process was based on manual rules maximizing the recall, in order to extract all forms prone to be abbreviations. Then, we filtered out all forms already in our list and manually checked the remaining ones, obtaining a final database with 2,022 entries.

3.2 Description of the platform

Leveraging the NLP analysis described above, the AMesure platform provides four types of diagnoses about texts to its users, as illustrated in Figure 1. The first diagnosis (marked by the letter A in the Figure 1) is a global readability score for the text. It is computed by a readability formula, specialized for administrative texts, that we previously developed (François et al., 2014). The output score ranges from 1 (for very easy texts) to 5 (for very complex texts) and is yielded by a support vector machine classifier combining 10 linguistic features of the text (e.g. word frequency, proportion of complex words, type-token ratio, mean length of sentence, ratio of past participle forms, etc.).

The second type of diagnosis (letter B in Figure 1) is more detailed and includes 11 readability yardsticks, each corresponding to one linguistic characteristic of the text known to affect reading. The psycholinguistic rationales for the choice of

these yardsticks have been discussed in length in François (2011), who has defined a set of 344 variables. Among this set, we have retained 11 yardsticks based on a correlational analysis (François et al., 2014). In the interface, the yardsticks are organised according to three linguistic dimensions of texts: lexicon, syntax, and textual aspects. The five lexical yardsticks capture (1) the percentage of difficult words, based on the list of 8000 simplest words in French (Gougenheim et al., 1964); (2) the number of rare words (see Section 3.1); (3) the density of abbreviations (see Section 3.1); (4) the proportion of unexplained abbreviations; and (5) the number of technical words (see Section 3.1). The four syntactic yardsticks include (1) the difficulty of the syntactic structures estimated roughly as the ratio of conjunctions and pronouns; (2) the mean number of words per sentence; (3) the ratio of structures considered as complex by plain language guides among all syntactic structures detected (see Section 3.1); and (4) the total number of sentences. As regards the two textual yardsticks, they include (1) a score corresponding to the level of personalization of the texts (text using pronouns at the first or at the second person are considered to be more readable (Daoust et al., 1996)); and (2) a score corresponding to the average intersentential coherence of the text. It is measured as the average cosine score between all adjacent sentences of the text, each of them being represented as a vector in a latent space (Foltz et al., 1998).

To render all these yardsticks more visual and more understandable, we project each of them on a five-degree scale, represented by colored feathers. The more feathers a yardstick gets, the more complex this linguistic dimension is supposed to be for reading. To transform the yardstick values into a five-degree scale, we applied the following method. Our corpus of 115 administrative texts has been annotated by experts on a five-degree difficulty scale (François et al., 2014). For each of our 11 yardsticks, we then estimated its Gaussian distribution (mean and standard deviation) on the corpus for each of the five levels. At running time, we simply compute the probability of the yardstick score for a given text to be generated by each of these five Gaussians and assign it the level corresponding to the higher probability.

The third type of diagnosis allows to directly visualize the text in which all complex phenomena annotated during the analysis step (see Section 3.1)

Figure 1: Result of a text analysis in AMesure.

are underlined, namely the three types of subordinated clauses, passives, parentheticals, rare words, abbreviations, and technical terms. For each of these categories, AMesure allows the user to select a tab showing only the respective phenomenon. It also offers a global view of the text in which complex sentences are highlighted in various shades of yellow (see letter C in Figure 1): the darker the yellow, the more difficult the sentence is to read.

Finally, the last functionality offers writing advice related to the complex phenomena detected (letter D in Figure 1). Two forms of advice are provided. On the one hand, we apply a list of 7 rules to filter out syntactic structures detected during the NLP analysis that should not be considered as complex. For instance, infinitive, participial, or even object clauses can be very short (e.g. *quand on décide d'**avoir un bébé*** or *le logement qu'**il occupe***) and are therefore not at all a burden for reading. The filtering rules were defined based on writing guidelines from three plain language guides for French (Gouvernement du Québec, 2006; Ministère de la Communauté française de Belgique, 2010; European Union, 2011). We also extracted

from these guides some pieces of advice that are shown to users of the platform when a difficult syntactic phenomenon is detected. Examples of advice are: "This sentence has 50 words. Please avoid sentences longer than 15 words" or "This sentence has three subordinate clauses. Please avoid having so many subordinate clauses in a sentence". On the other hand, we also offer simpler synonyms for words detected as rare words or technical words. The synonyms are taken from ReSyf (Billami et al., 2018), a lexical resource in which synonyms are ranked by difficulty. For now, we show the three simpler synonyms found in ReSyf for a given difficult word, letting the user to pick the best one. More advanced methods based on embeddings are, however, considered at the moment to improve the automatic selection.

4 Evaluation of the system

To assess the performance of the various extraction algorithms included in our platform, three linguists manually annotated, in 24 administrative texts, the following five phenomena: passive structures, relative clauses, object clauses, adverbial clauses, and abbreviations[6]. The work of annotators was supported by guidelines focusing on difficult cases[7]. At the end of the annotation process, the expert agreement was evaluated using Fleiss' kappa (see Table 1). The agreement was high for the rather easy tasks of annotating abbreviations and passive clauses. Detecting subordinate clauses is, however, a much more complex task, if only because it is also necessary to identify the type of structures. A common reference version was then built through consensus-building.

This gold-standard version of the annotation was manually compared to the output of AMesure for the 24 texts in the test set. Table 1 reports the results of this evaluation in terms of recall, precision, and F1-score for the different types of structures. Performance for the detection of passive clauses, relative clauses, adverbial clauses and abbreviations are satisfactory (F1 is above .8). By comparison, Zilio et al. (2017), who detect syntactic structures in English, obtained a precision of 0.88

Phenomena	**R**	**P**	**F1**	κ
Passive clauses	0.92	0.92	0.92	0.92
Subordinated clauses (all)	0.84	0.87	0.85	0.47
Relative clauses	0.83	0.88	0.86	/
Object clauses	0.56	0.42	0.48	/
Adverbial clauses	0.78	0.83	0.8	/
Abbreviations	0.73	0.9	0.8	0.97
Total (macro-average)	0.83	0.9	0.86	0.79

Table 1: Recall (R), precision (P), F1, percentage of agreement and Fleiss' κ scores for the five phenomena detected in the platform.

and a recall of 0.62 for the relative clauses and a recall of 0.66 and a precision of 0.94 for infinitive clauses introduced by the particle "TO". Chinkina and Meurers (2016) reached a recall of 0.83 and a precision of 0.71 for relative clauses. However, our system has trouble detecting object clauses, which have a F1-score of only 0.48. Investigation of the confusion matrix reveals that 77% of object clauses (37 out of 48) are correctly detected by AMesure, but 17 out of 37 are wrongly classified as adverbial clauses. This is a limited issue, as advice can still be provided even if the system gets the type of clause wrong.

5 Conclusion

We have presented the AMesure system, which automatically analyzes the readability of French administrative texts based on classic readability metrics, but also on guidelines from plain language books. The system is freely available through a web platform and is aimed to help writers of administrative texts to produce more accessible documents and forms. To that purpose, it offers a global readability score for the texts, 11 readability yardsticks, a detailed diagnosis in which difficult linguistic words and syntactic structures are highlighted, and some plain language advice. We also carried out a manual evaluation of the system based on 24 administrative texts annotated by linguists. Performance is satisfactory, except as regards the identification of object clauses. More work is needed on this category, especially to distinguish it from adverbial clauses. We also plan to improve the system providing simpler synonyms by adding a semantic filter based on embedding models. Finally, we plan to conduct a study with real writers of administrative texts to measure the perceived usefulness of AMesure as a whole, but also the usefulness of each functionality.

[6]The detection of rare words and complex technical terms could not be assessed according to the same protocol as what matters is the psychological relevance of their identification for a given audience of readers. Further experiments with subjects are required to assess these two dimensions.

[7]For instance, infinitive clauses led by a semi-modal auxiliary such as *devoir* (ought to) or *pouvoir* (can) were discussed, as contradictory points of view can be found in grammars.

Acknowledgments

We would like to thank the "Direction de la language française" from the Federation Wallonia-Brussels for its continued support to the AMesure project since 2014. We also want to acknowledge the wonderful work of Romain Pattyn, Gaëtan Ansotte, Baptiste Degryse on the interface and the great visuals of Brian Delmée.

References

M. Adler. 2012. The plain language movement. In *The Oxford handbook of language and law*.

R. Alarcon, L. Moreno, I. Segura-Bedmar, and P. Martinez. 2019. Lexical simplification approach using easy-to-read resources. *Procesamiento de Lenguaje Natural*, 63:95–102.

M. Billami, T. François, and N. Gala. 2018. ReSyf: A French lexicon with ranked synonyms. In *Proceedings of COLING 2018*.

S. Bott, L. Rello, B. Drndarevic, and H. Saggion. 2012. Can Spanish Be Simpler? LexSiS: Lexical Simplification for Spanish. pages 357–374.

M. Candito, J. Nivre, P. Denis, and E. H. Anguiano. 2010. Benchmarking of statistical dependency parsers for French. In *Proceedings of COLING 2010*, pages 108–116.

R. Catherine. 1968. *Le style administratif*. A. Michel.

R. Chandrasekar, C. Doran, and B. Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the 16th conference on Computational Linguistics*, volume 2, pages 1041–1044.

M. Chinkina and D. Meurers. 2016. Linguistically aware information retrieval: Providing input enrichment for second language learners. In *Proceedings of BEA 2016*, pages 188–198.

T.M. Chung. 2003. A corpus comparison approach for terminology extraction. *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication*, 9(2):221–246.

W. Coster and D. Kauchak. 2011. Simple English Wikipedia: A new text simplification task. In *Proceedings of ACL 2011*, pages 665–669. Association for Computational Linguistics.

F. Cusin-Berche. 2003. *Les mots et leurs contextes*. Presses Sorbonne nouvelle.

M. Cutts. 2020. *Oxford guide to plain English*. Oxford University Press, USA.

F. Daoust, L. Laroche, and L. Ouellet. 1996. SATO-CALIBRAGE: Présentation d'un outil d'assistance au choix et à la rédaction de textes pour l'enseignement. *Revue québécoise de linguistique*, 25(1):205–234.

J. De Belder and M.-F. Moens. 2010. Text simplification for children. In *Proceedings of the SIGIR workshop on accessible search systems*, pages 19–26.

P. Denis and B. Sagot. 2012. Coupling an annotated corpus and a lexicon for state-of-the-art POS tagging. *Language resources and evaluation*, 46(4):721–736.

K. Desbiens. 2008. Les obstacles à la simplification: le cas des membres du centre d'expertise des grands organismes. *Langue, médiation et efficacité communicationnelle. Québec*.

European Union. 2011. *How to write clearly* . Publications Office of the EU, Luxembourg.

J. Falkenjack, E. Rennes, D. Fahlborg, V. Johansson, and A. Jönsson. 2017. Services for text simplification and analysis. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 309–313.

P.W. Foltz, W. Kintsch, and T.K. Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse processes*, 25(2):285–307.

T. François. 2011. *Les apports du traitement automatique du langage à la lisibilité du français langue étrangère*. Ph.D. thesis, Université Catholique de Louvain. Thesis Supervisors : Cédrick Fairon and Anne Catherine Simon.

T. François, L. Brouwers, H. Naets, and C. Fairon. 2014. AMesure: une formule de lisibilité pour les textes administratifs. In *Proceedings of TALN 2014*.

T. François, A. Müller, B. Degryse, and C. Fairon. 2018. Amesure : une plateforme web d'assistance à la rédaction simple de textes administratifs. *Repères DoRiF*, 16(1).

G. Glavaš and S. Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of ACL-IJCNLP 2015*, pages 63–68.

G. Gougenheim, R. Michéa, P. Rivenc, and A. Sauvageot. 1964. *L'élaboration du français fondamental (1er degré)*. Didier, Paris.

Gouvernement du Québec. 2006. *Rédiger simplement - Principes et recommandations pour une langue administrative de qualité* . Bibliothèques et archives nationales du Québec, Québec.

J. Kimble. 1992. Plain english: A charter for clear writing. *TM Cooley L. Rev.*, 9:1.

J. Kimble. 1996. Writing for dollars, writing to please. *Scribes J. Leg. Writing*, 6:1.

B. Labasse. 2001. L'institution contre l'auteur : pertinence et contraintes en rédaction professionnelle. In *Congrès annuel de l'ACPRST/CATTW, Université Laval, Québec*.

J. Lee, W. Zhao, and W. Xie. 2016. A customizable editor for text simplification. In *Proceedings of COLING 2016: System Demonstrations*, pages 93–97.

R. Levy and G. Andrew. 2006. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of LREC 2006*, pages 2231–2234.

B. Madinier. 2009. Langage administratif et modernisation de l'état : pour une communication réussie. In *La communication avec le citoyen : efficace et accessible ? Actes du colloque de Liège 2009*, pages 55–66.

Ministère de la Communauté française de Belgique. 2010. *Écrire pour être lu - Comment rédiger des textes administratifs faciles à comprendre ?* Ingber, Damar, Bruxelles.

B. New, M. Brysbaert, J. Veronis, and C. Pallier. 2007. The use of film subtitles to estimate word frequencies. *Applied Psycholinguistics*, 28(04):661–677.

S. Nisioi, S. Štajner, S. P. Ponzetto, and L.P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of ACL2017: Short Papers)*, pages 85–91.

A. Nord. 2018. Plain language and professional writing : A research overview. Technical report, Language Council of Sweden.

OECD. 2016. *Skills Matter : Further Results from the Survey of Adult Skills*. OECD Publishing, Paris.

G. Paetzold and L. Specia. 2016. Unsupervised lexical simplification for non-native speakers. In *Thirtieth AAAI Conference on Artificial Intelligence*.

G. Paetzold and L. Specia. 2017. Lexical simplification with neural ranking. In *Proceedings of EACL2017: Short Papers*, pages 34–40.

Plain Language Action and Information Network. 2011. Federal plain language guidelines.

J. Qiang, Y. Li, Y. Zhu, Y. Yuan, and X. Wu. 2020. Lexical simplification with pretrained encoders. In *Proceedings of AAAI 2020*, pages 8649–8656.

H. Saggion. 2017. Automatic text simplification. *Synthesis Lectures on Human Language Technologies*, 10(1):1–137.

C. Scarton, M. Oliveira, A. Candido Jr, C. Gasperin, and S. Aluísio. 2010. Simplifica: a tool for authoring simplified texts in brazilian portuguese guided by readability assessments. In *Proceedings of the NAACL HLT 2010: Demonstration Session*, pages 41–44.

M. Shardlow. 2014. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70.

A. Siddharthan. 2011. Text simplification using typed dependencies: A comparison of the robustness of different generation strategies. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 2–11.

A. Siddharthan. 2014. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298.

M. da Silva Conrado, A. Di Felippo, T.A.S. Pardo, and S.O. Rezende. 2014. A survey of automatic term extraction for brazilian portuguese. *Journal of the Brazilian Computer Society*, 20(1):12.

S. Sinha, F. McDermott, G. Srinivas, and P.W.J. Houghton. 2011. Use of abbreviations by healthcare professionals: what is the way forward? *Postgraduate medical journal*, 87(1029):450–452.

L. Specia. 2010. Translating from Complex to Simplified Sentences. In *Proceedings of Propor 2010.*, pages 30–39.

S.M. Yimam and C. Biemann. 2018. Demonstrating par4sem-a semantic writing aid with adaptive paraphrasing. In *Proceedings of EMNLP 2018: System Demonstrations*, pages 48–53.

X. Zhang and M. Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of EMNLP 2017*, pages 584–594.

Z. Zhu, D. Bernhard, and I. Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1353–1361. Association for Computational Linguistics.

L. Zilio, R. Wilkens, and C. Fairon. 2017. Using nlp for enhancing second language acquisition. In *Proceedings of RANLP 2017*, pages 839–846.

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

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.
- **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.

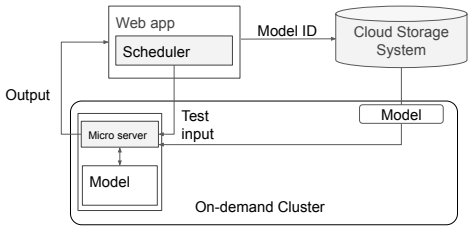

Figure 1: AUTONLU architecture. In the figure is the dataflow when the user calls to the `/test` endpoint.

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

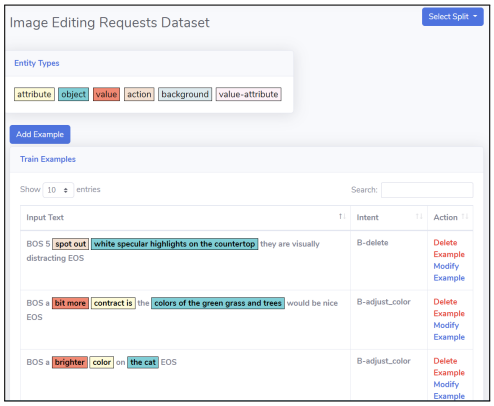

Figure 2: Dataset view of AUTONLU.

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.

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

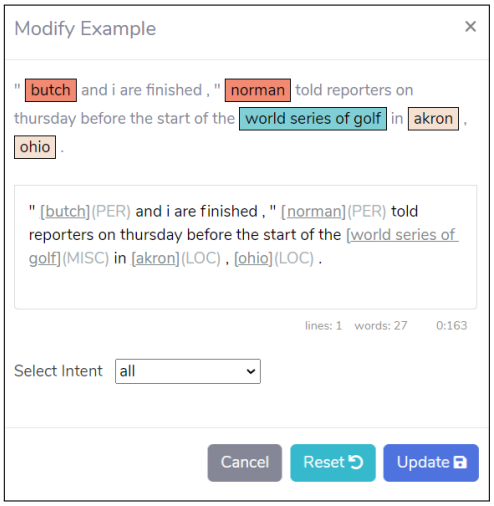

Figure 3: Edit/Add a datapoint.

Figure 4: An example interactive confusion matrix.

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 $y_i \in \{\texttt{B}, \texttt{I}, \texttt{O}\}$. 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.

ISA: An Intelligent Shopping Assistant

Tuan Manh Lai [*1,2], **Trung Bui** [2], **Nedim Lipka** [2]
[1] University of Illinois at Urbana-Champaign
[2] Adobe Research

Abstract

Despite the growth of e-commerce, brick-and-mortar stores are still the preferred destinations for many people. In this paper, we present ISA, a mobile-based intelligent shopping assistant that is designed to improve shopping experience in physical stores. ISA assists users by leveraging advanced techniques in computer vision, speech processing, and natural language processing. An in-store user only needs to take a picture or scan the barcode of the product of interest, and then the user can talk to the assistant about the product. The assistant can also guide the user through the purchase process or recommend other similar products to the user. We take a data-driven approach in building the engines of ISA's natural language processing component, and the engines achieve good performance.

1 Introduction

Shopping in physical stores is a popular option for many people. Each week, a lot of people enter supermarkets in which they are immersed with many different product choices. In many shopping centers, customer service representatives (CSRs) are employed to answer questions from customers about products. However, a customer may experience long waiting time for assistance if all CSRs are busy interacting with other customers. Therefore, automated solutions can increase customer satisfaction and retention.

In this paper, we introduce a mobile-based intelligent shopping assistant, ISA, which is based on advanced techniques in computer vision, speech processing, and natural language processing. A user just needs to take a picture or scan the barcode of the product of interest. After that, the user can ask ISA a variety of questions such as product features, specifications and return policies. The assistant can also guide the user through the purchase process or recommend other similar products. This work can be used as the first step in fully automating customer service in shopping centers. With ISA, no CSRs will be needed as customers can simply turn to their phones for assistance. We have developed a fully functional prototype of ISA.

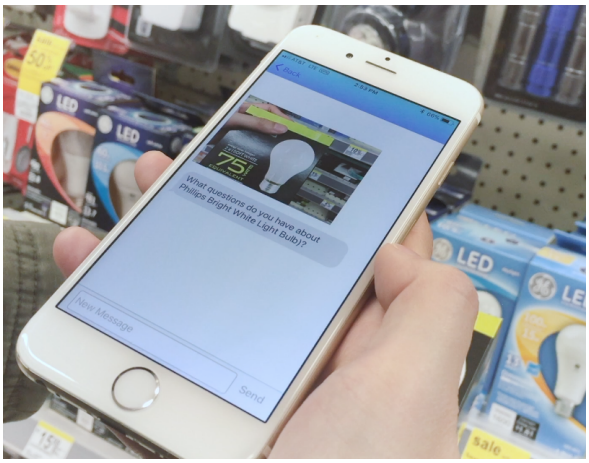

Figure 1: ISA assists users at physical stores

The rest of the paper is organized as follows. Section 2 introduces some related work. Section 3 gives an overview of the design and implementation of the system. Finally, Section 4 concludes the paper and suggests future directions.

2 Related Work

The most closely related branches of work to ours are probably customer service chatbots for e-commerce websites. For example, SuperAgent (Cui et al., 2017) is a powerful chatbot that leverages large-scale and publicly available e-commerce data. The researchers demonstrate SuperAgent as an add-on extension to mainstream web browsers. When a user visits a product page, SuperAgent crawls the information of the product from multi-

[1] The work was conducted while the first author 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 14–19

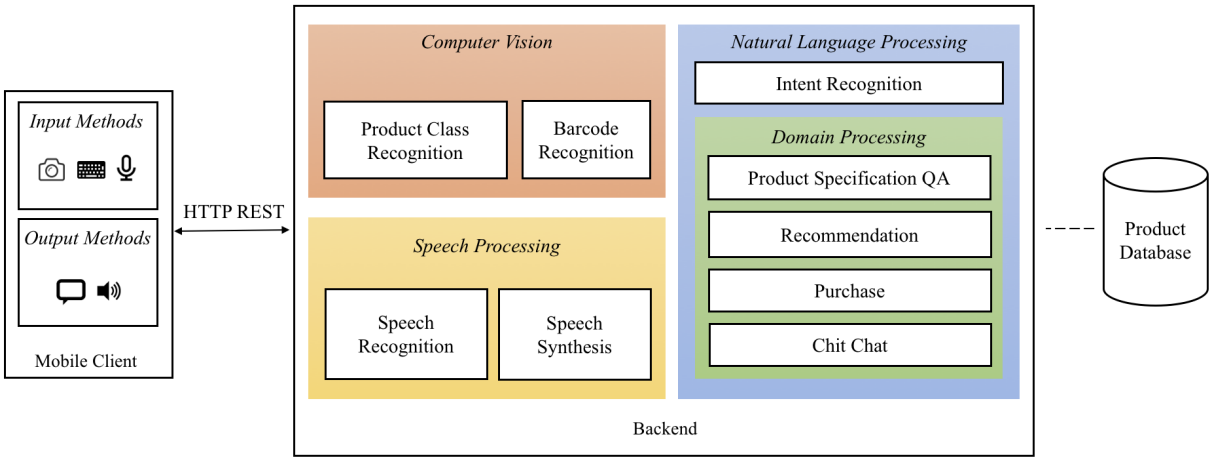

Figure 2: The system overview of ISA

ple data sources within the page. After that, the user can ask SuperAgent about the product. Unlike SuperAgent, ISA is designed to assist users at physical stores (Figure 1). In addition to natural language processing techniques, ISA also needs to use techniques in computer vision and speech processing when interacting with the users.

3 System Description

3.1 Overview

When an in-store user wants to get more information about a specific product, the user just needs to take a picture or scan the barcode of the product. The system then retrieves the information of the product of interest from a database by using computer vision techniques. After that, the user can ask natural language questions about the product specifications to the system. The user can either type in the questions or directly speak out the questions using voice. ISA is integrated with both speech recognition and speech synthesis abilities, which allows users to ask questions without typing.

Figure 2 shows the system overview of ISA. As the figure shows, a mobile client communicates with the backend through a well-defined HTTP REST API. This creates a separation between the client and the server, which allows ISA to be scaled without much difficulty. The backend consists of three main components: 1) speech processing, 2) computer vision, 3) natural language processing. Users can chat with ISA in speech. The speech recognition and speech synthesis are implemented by calling third-party services. The computer vision component is responsible for recognizing the products that the user is facing. Given an image of a product of interest, a fine-grained visual object classification model will be used to identify the product and retrieve its information. This task is challenging because many products are visually very similar (e.g., washers and dryers usually have similar shape). Therefore, we enhance the component with highly accurate standard algorithms for barcode recognition. In case it is difficult for the object classification model to identify the product of interest accurately, the user can simply scan the barcode of the product. Finally, the natural language processing component is responsible for generating a response from a text query or question. We will next detail each part of the natural language processing component in the following sections.

Intent Types	Example Query
Product Specification QA	How heavy is this chair?
Recommendation	Show me some other items
Purchase	I want to buy this.
Chit Chat	How are you doing?

Table 1: Intent Types

3.2 Intent Recognition

When ISA receives a query from a user, the intent recognition engine is used to determine the intent of the query. Based on the recognized intent, the appropriate domain-specific engine will be triggered. We define four different types of intent as shown in Table 1. Intent detection can be naturally treated as a classification problem. In this work we build a random forest model (Breiman, 2001) for the problem and it achieves good performance. Other popular classifiers like support vector machines (Haffner et al., 2003) and deep

Figure 3: Answering questions regarding product specifications

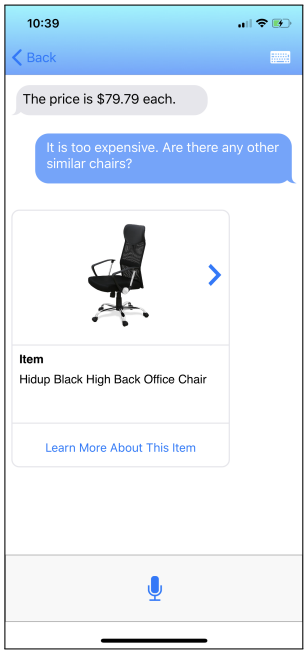

Figure 4: ISA recommends similar products to the user

neural network methods (Sarikaya et al., 2011) can also be applied in this case.

We create a dataset of 500 different queries and use it to build a random forest (RF) for intent classification. Approximately 2/3 of the cases are used as training set, whereas the rest (1/3) are used as test set, in order to estimate the model's performance. We create a bag-of-words feature vector for each query and use it as input for the RF. The number of trees in the forest is set to be 80. For each node split during the growing of a tree, the number of features used to determine the best split is set to be $\sqrt{k}$ where k is the total number of features of the dataset. The accuracy of the trained RF model evaluated on the test set is 98.20%.

3.3 Product Specification QA

The product specification QA engine is used to answer questions regarding the specifications of a product. For every product, there is a list of specifications in the form of (specification_name, specification_value). We formalize the task of the engine as follows: Given a question Q about a product P and the list of specifications $(s_1, s_2, ..., s_M)$ of P, the goal is to identify the specification that is most relevant to the question Q. M is the number of specifications of the product, and s_i is the sequence of words in the name of the i^{th} specification. In this formulation, the task is similar to the answer selection problem. 'Answers' shall be individual product specifications.

Previous methods for answer selection typically relies on feature engineering, linguistic tools, or external resources (Wang and Manning, 2010; Heilman and Smith, 2010; Yih et al., 2013; Yao et al., 2013). Recently, with the renaissance of neural network models, many deep learning based methods have been proposed to tackle the answer selection problem (Rao et al., 2016; Zhiguo Wang, 2017; Bian et al., 2017; Shen et al., 2017; Tran et al., 2018; Lai et al., 2018a; Tay et al., 2018; Lai et al., 2018b,c; Rao et al., 2019; Lai et al., 2019; Garg et al., 2019; Kamath et al., 2019; Laskar et al., 2020). These deep learning based methods typically outperform traditional techniques without relying on any feature engineering or expensive external resources. For example, the IWAN model proposed in (Shen et al., 2017) achieves competitive performance on public datasets such as TrecQA (Wang et al., 2007) and WikiQA (Yang et al., 2015).

Using Amazon Mechanical Turk, a popular crowdsourcing platform, we create a dataset of 6,922 questions that are related to 369 specifications and 148 products listed in the Home Depot website. We implement the IWAN model and train the model on the collected dataset. The top-1 accuracy, top-2 accuracy, and top-3 accuracy of the model evaluated on a held-out test set are 85.60%, 95.80%, and 97.60%, respectively.

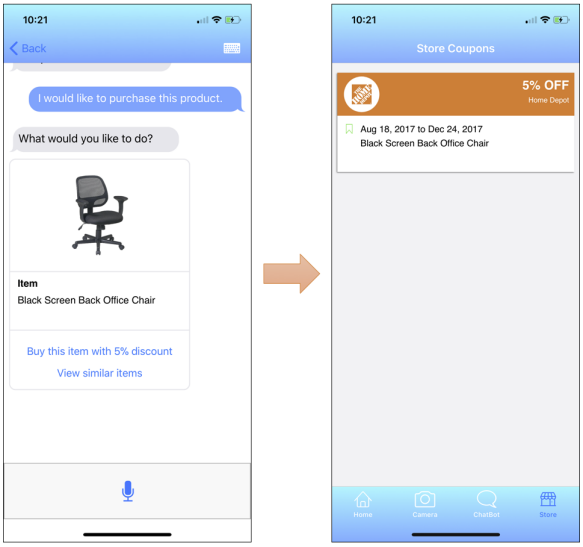

Figure 5: The user purchased an office chair with 5% discount

In production, given a question about a product, the trained model is used to rank every specification of the product based on how relevant the specification is. We select the top-ranked specification and use it to generate the response sentence using predefined templates (Cui et al., 2017). An example of the product specification QA engine's outputs is shown in Figure 3. The first question from the user is matched to the product weight specification, whereas the second question is matched to the return policy specification.

3.4 Recommendation

The recommendation engine is responsible for giving new suggestions and recommendations to users. When a user wants to look for similar products (e.g., by saying "Are there any other similar products?"), the engine will search the database for related products and then send the information of them to the app for displaying to the user (Figure 4).

3.5 Purchase

The purchase engine is responsible for guiding the user through the purchase process. When a user wants to buy a specific product (e.g., by saying "I would like to purchase this product."), the engine will first query the database for information such as the product listing price, available discounts, and user payment information. After that, the engine will craft a special response message and send it to the client app in the user's mobile device. The response message will instruct the app how to assist the user through the purchase process or provide personalized discounts if applicable (Figure 5).

3.6 Chit Chat

The chit chat engine is used to reply to greeting queries such as "How are you doing?" or queries that are off the subject such as "Is the sky blue?". Our approach to building the engine is based on the sequence-to-sequence (seq2seq) framework (Sutskever et al., 2014). The model consists of two recurrent neural networks: an encoder and a decoder. The encoder converts the input query into a fixed size feature vector. Based on that feature vector, the decoder generates the output response, one word at a time. The model is integrated with the global attention mechanism (Luong et al., 2015) so that the decoder can attend to specific parts of the input query when decoding instead of relying only on the fixed size feature vector. We collect about 3M query-response pairs from Reddit and use them to train the seq2seq model. Examples of the engine's outputs are shown below:

Q: How are you doing?
A: I'm doing well.
Q: Is the sky blue?
A: Yes.

4 Conclusion and Future Work

In this paper, we present ISA, a powerful intelligent shopping assistant. ISA is designed to achieve the goal of improving shopping experience in physical stores by leveraging advanced techniques in computer vision, speech processing, and natural language processing. A user only needs to take a picture or scan the barcode of the product of interest, and then the user can ask ISA a variety of questions about the product. The system can also guide the user through the purchase decision or recommend other similar products to the user.

There are many fronts on which we will be exploring in the future. Currently the product specification QA engine answers only questions regarding the specifications of a product. We will implement engines for addressing other kinds of questions. We will also extend ISA to better support other languages and informal text (Nguyen and Nguyen, 2020; Nguyen et al., 2020; Martin et al., 2020). In addition, we will conduct a user study to evaluate our system in the future. Finally, we wish to extend this work to other domains such as building an as-

sistant for handling image editing requests (Brixey et al., 2018).

5 Acknowledgments

The authors wish to thank Dr. Hung Bui (VinAI Research) and Dr. Sheng Li (University of Georgia) for their guidance and feedback on this project.

References

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 1987–1990.

Leo Breiman. 2001. Random forests. *Mach. Learn.*, 45(1):5–32.

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*.

Lei Cui, Furu Wei, Shaohan Huang, Chuanqi Tan, Chaoqun Duan, and Ming Zhou. 2017. Superagent: A customer service chatbot for e-commerce websites. In *Proceedings of ACL 2017, System Demonstrations*, pages 97–102. Association for Computational Linguistics.

Siddhant Garg, Thuy Vu, and Alessandro Moschitti. 2019. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. *arXiv preprint arXiv:1911.04118*.

P. Haffner, G. Tur, and J. H. Wright. 2003. Optimizing svms for complex call classification. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 1, pages I–632–I–635 vol.1.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 1011–1019, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sanjay Kamath, B. Grau, and Y. Ma. 2019. Predicting and integrating expected answer types into a simple recurrent neural network model for answer sentence selection. *Computación y Sistemas*, 23.

Tuan Lai, Trung Bui, Sheng Li, and Nedim Lipka. 2018a. A simple end-to-end question answering model for product information. In *Proceedings of the First Workshop on Economics and Natural Language Processing*, pages 38–43, Melbourne, Australia. Association for Computational Linguistics.

Tuan Lai, Trung Bui, Nedim Lipka, and Sheng Li. 2018b. 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. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5953–5959, Hong Kong, China. Association for Computational Linguistics.

Tuan Manh Lai, Trung Bui, and Sheng Li. 2018c. 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.

Md Tahmid Rahman Laskar, Jimmy Xiangji Huang, and Enamul Hoque. 2020. Contextualized embeddings based transformer encoder for sentence similarity modeling in answer selection task. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5505–5514, Marseille, France. European Language Resources Association.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Dat Quoc Nguyen and Anh Tuan Nguyen. 2020. PhoBERT: Pre-trained language models for Vietnamese. *Findings of EMNLP*.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916. ACM.

Jinfeng Rao, Linqing Liu, Yi Tay, Wei Yang, Peng Shi, and Jimmy Lin. 2019. Bridging the gap between relevance matching and semantic matching for short text similarity modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5370–5381, Hong Kong, China. Association for Computational Linguistics.

R. Sarikaya, G. E. Hinton, and B. Ramabhadran. 2011. Deep belief nets for natural language call-routing. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5680–5683.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1190–1200.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-cast attention networks. In *KDD*.

Quan Hung Tran, Tuan Lai, Gholamreza Haffari, Ingrid Zukerman, Trung Bui, and Hung Bui. 2018. The context-dependent additive recurrent neural net. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1274–1283, New Orleans, Louisiana. Association for Computational Linguistics.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1164–1172, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Xuchen Yao, Benjamin Van Durme, Chris Callison-burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1744–1753, Sofia, Bulgaria. Association for Computational Linguistics.

Radu Florian Zhiguo Wang, Wael Hamza. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4144–4150.

metaCAT: A Metadata-based Task-oriented Chatbot Annotation Tool

Ximing Liu, Wei Xue, Qi Su, Weiran Nie, Wei Peng *
Huawei Technologies Co., Ltd., PRC
{liuximing1,xuewei5,qi.su,nieweiran,peng.wei1}@huawei.com

Abstract

Creating high-quality annotated dialogue corpora is challenging. It is essential to develop practical annotation tools to support humans in this time-consuming and error-prone task. We present metaCAT, which is an open-source web-based annotation tool designed specifically for developing task-oriented dialogue data. To the best of our knowledge, metaCAT is the first annotation tool that provides comprehensive metadata annotation coverage to the domain, intent, and span information. The data annotation quality is enhanced by a real-time annotation constraint-checking mechanism. An Automatic Speech Recognition (ASR) function is implemented to allow users to paraphrase and create more diversified annotated utterances. metaCAT is publicly available for the community. [1]

1 Introduction

The progress of the development of multi-turn task-oriented dialogue systems has been largely constrained by the availability of large-scale high-quality data (Zhu et al., 2020). Due to the complexity of annotating task-oriented dialogue data, a large number of errors have been found in existing benchmark datasets, e.g., MultiWOZ (Budzianowski et al., 2018). Despite ongoing efforts such as re-annotating the state tags based on the original utterances in MultiWOZ 2.1 (Eric et al., 2019) and introducing slot span annotations in MultiWOZ 2.2 (Zang et al., 2020), there is still much room for improvement. Creating high-quality annotated dialogue corpora is highly challenging thus necessitates a high level of human engagements. It is essential to develop practical annotation tools for supporting experts in this time-consuming and error-prone task.

Although many NLP annotation tools exist, LIDA (Collins et al., 2019) is the only one designed specifically for annotating multi-turn task-oriented benchmark dialogue datasets like MultiWOZ. As a lightweight interactive dialogue annotator, LIDA provides an end-to-end pipeline for converting raw text to structured conversation data. It claims to support machine learning-based annotation recommenders and provide an interface resolving inter-annotators disagreements. LIDA is an easy-to-use tool for tagging Boolean and String type slot values. However, it lacks functionality for annotating complex data. These data include enumeration, span information, and multi-valued slot values for utterance and other metadata such as domains, intents, etc.

A critical step in ensuring data quality when annotating is to perform real-time constraint-checking. Retrospective inter-annotator disagreement resolution (i.e., after an annotation task is completed) is an option offered by LIDA. It is desirable to design a function which provides real-time constraint-checking to eliminate human errors. For example, the slot value "dontcare" should be avoided in the system's annotation, because it can only be applied to a user's annotations. This highlights the need to introduce metadata to define the scope of the annotation and, in the meantime, enforce consistency between human inputs and underlying constraints.

Data annotation task is time-consuming as there are large amounts of typing actions involved. To improve the efficiency of the annotation task, we identify the need to integrate the Automatic Speech Recognition (ASR) function to the annotation process. By automatically converting speech to text, ASR can speed up the data input process otherwise done by keyboard typing. ASR can also be used to create paraphrases of utterances, and as a consequence, enhance the diversity of the spoken

*Corresponding author
[1] https://github.com/lexmen318/metaCAT

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 20–25

language. This is especially useful in scenarios where manually collected dialogue corpora are rare and expensive. Providing such diversity is regarded as the key to increasing the robustness of dialogue models trained with these data.

In this paper, we propose metaCAT, which is a web-based task-oriented chatbot annotation tool with complete management of users, tasks, datasets, and dialogues. metaCAT extends LIDA by contributing additional key useful features including:

- comprehensive metadata annotation coverage to the domain, intent and span information w.r.t. each dialogue turn;
- real-time annotation constraint-checking to ensure data quality;
- ASR paraphrasing to speed up an annotator's data input process and increase the diversity of utterances.

2 Related Work

Various annotation tools have been developed, targeting NLP tasks in recent years. As depicted in Collins et al. (2019), these tools are built for disparate NLP tasks ranging from general text processing (i.e., GATE (Cunningham et al., 2002)) to entity linking (e.g., INCEpTION (Klie et al., 2018)). Among them, a number of dialogue annotation tools are discussed below.

DialogueView (Yang and Heeman, 2005) is a dialogue annotation tool developed for segmenting recorded dialogue into utterances, annotating speech repairs, tagging speech acts, and segmenting dialogue into hierarchical discourse segments. The focus is on dialogue segmentation; therefore features for generating task-oriented dialogue data, for example slot-value labeling, are not available.

TWIST (Pluss, 2012) supports turn segmentation and content feature annotation. After highlighting and creating new turn segments, users can assign predefined content features to the targeted turn segment. The content feature includes descriptions like "Objective", "Subjective", etc. However, there is no means to support label classification and slot-value annotation.

DART (Weisser, 2016) stands for Dialogue Annotation and Research Tool, which is a research environment enabling users to annotate and analyze dialogues automatically. DART was developed for linguistic research and analysis. DART provides a convenient means for leveraging linguistic resources to improve the annotation results and test hypotheses. Only intent and enumeration slot can be labeled via this tool. It is not designed to create task-oriented dialogue data for training dialogue models, therefore lacking most of the related features.

LIDA (Collins et al., 2019) claims to be the first annotator capable of providing a full end-to-end dialogue annotation pipeline. It is a web-based tool designed specifically for task-oriented dialogue systems. It provides basic features to annotate the Boolean or String type slot-value of each turn of the dialogue. LIDA integrates with back-end machine learning models as annotation recommenders to achieve semi-automatic annotating. It contains a dedicated interface to resolve inconsistent annotations between different annotators. Despite implementing several key features for annotating task-oriented dialogue data, LIDA lacks functionality supporting users to handle complex data annotation, such as enumeration, span information and multi-valued slot labeling. LIDA can only handle inter-annotator disagreement resolution at the post-editing stage. Real-time annotation error-checking is out of the scope.

In this paper, we present metaCAT, which is designed to fill the above gaps. Table 1 provides a detailed comparison of the features and capabilities provided by metaCAT and other dialogue annotation tools.

3 System Overview

metaCAT consists of a front-end and a back-end. The back-end is implemented with Flask providing the RESTful service for data manipulation while the front-end is developed with the VUE.js framework. metaCAT uses MongoDB as a database.

The user interfaces for the administrator and annotators are developed independently as the roles and privileges are different. The administrator is responsible for uploading metadata and dialogue data, assigning and monitoring the annotation tasks, and downloading annotated data. Annotators are responsible for the execution of annotating tasks and paraphrasing tasks. All tasks are managed in batches with each containing several to dozens of dialogues. The intermediate results of annotation are saved to MongoDB.

metaCAT defines a comprehensive annotation ontology system for building task-oriented dia-

Features	metaCAT	LIDA	TWIST	DART
Classification Labels	YES	YES	NO	YES
Edit Dialogues/Turns	YES	YES	YES	YES
Turn/Dialogue Segmentation	YES	YES	YES	YES
Recommenders	YES	YES	NO	NO
Annotation Metadata Import	YES	NO	NO	NO
Comprehensive Tagging	YES	Boolean/String Slot	NO	Intent/Enum. Slot
Paraphrasing (via ASR or Keyboard)	YES	NO	NO	NO
Real-time Constraint-Checking	YES	NO	NO	NO
Comprehensive Management	YES	Dialogue Only	NO	Dialogue Only
Multilingual Support	YES	NO	NO	NO

Table 1: A detailed comparison of the features offered by metaCAT and other dialogue annotation tools. "Comprehensive Tagging" indicates if a tool provides annotation for a comprehensive range of items including Utterance/Domain/Intent/Slot/Span. "Comprehensive Management" refers to the management of a comprehensive range of items including User/Task/Dataset/Dialogue. "Enum." stands for enumeration.

logue data. We follow the way how SGD (Rastogi et al., 2019) organizes metadata:

- **General Domain**: The general domain contains intents with no slots, for example, "Thank" or "Bye";
- **Service Domain**: These domains involve specific intents with slots. An utterance usually belongs to one service domain, but in some rare cases it may contain two or more service domains, such as an utterance booking "Hotel" and "Taxi" at the same time;
- **Intent**: An utterance usually implies one or more specific intents.
- **Slot**: Slots generally represent key information carried with the intent.

The slots can be classified into enumeration type and string type. An enumeration slot contains a list of specific slot values without any span information. A string slot is usually one word or short segment of text taken from the utterance. Some special values of a string slot exist without span information, e.g., "dontcare". A slot usually corresponds to one value but there are some slots containing multiple values, such as names of two restaurants recommended by the system.

In addition to offering a comprehensive metadata annotation ontology, metaCAT also supports a dialogue paraphrasing feature, which requires the annotators to paraphrase the utterances via keyboard or ASR before the annotation task. Figure 1 depicts the journey of the administrator and annotators in using metaCAT. A short video clip is available in our YouTube channel [2] demonstrating the system.

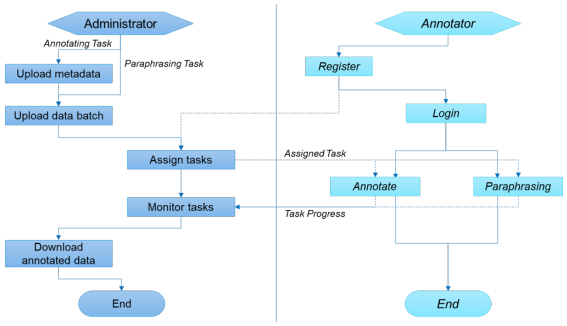

Figure 1: The journey of the administrator and annotators using metaCAT.

3.1 Administrator Journey

Before creating annotation tasks, the administrator needs to upload metadata of the dataset, which is generated offline based on a provided template file in JSON format. After the metadata is uploaded, dialogue data for annotating can be imported. Currently, metaCAT supports the following formats of dialogue data:

- **Raw Format**: It contains crowd-sourced raw utterances from both the user side and the system side without any annotation.
- **Annotating Format**: This is the same format as the output file of metaCAT. It includes both utterances and associated annotations. This is the format used for annotating in metaCAT, which may have been converted from formats used in other annotating tools.

[2] https://youtu.be/07_PWD4_c4E

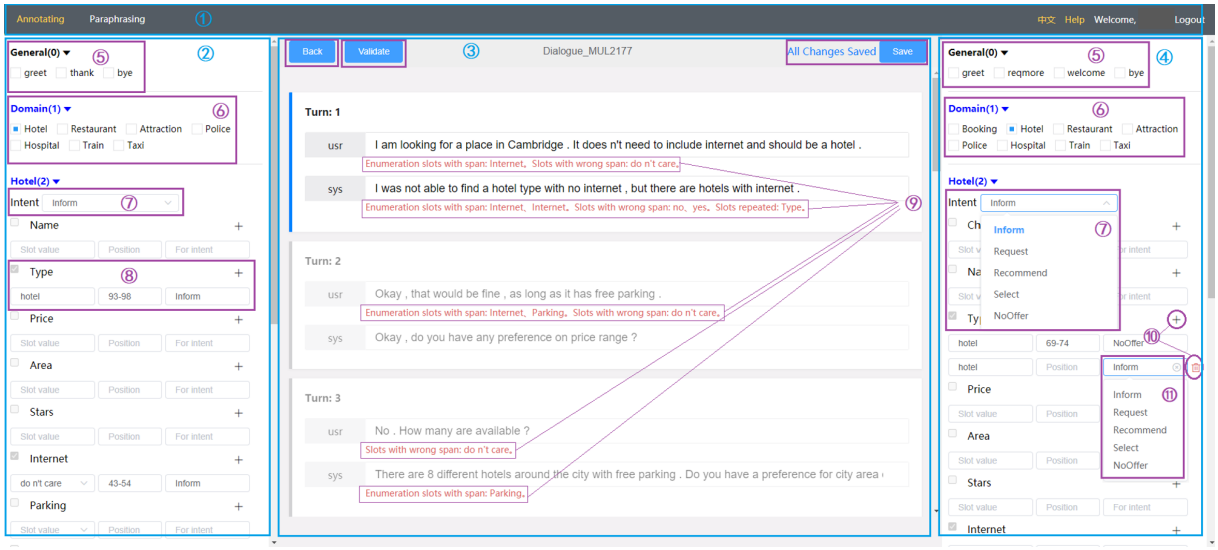

Figure 2: Annotating Interface - to annotate "usr" and "sys" utterance. Note: ①:menu bar; ②: "usr" annotating zone; ③ utterance zone; ④: "sys" annotating zone; ⑤: general intent selection; ⑥: service domain selection; ⑦: default intent selection; ⑧:slot-value editing box; ⑨:validating results; ⑩:slot-value adding/deletion; ⑪:slot attached intent modification.

- **Specific Format**: Only one specific format is supported currently, which is MultiWOZ format. It is automatically converted to metaCAT format after uploaded.

- **Paraphrasing Format**: It applies to the paraphrased utterances and related annotations. Data format conversion is performed off-line before paraphrasing.

The result data can be exported in three types of format: annotating format, MultiWOZ format, and paraphrasing format.

3.2 Annotator Journey

After registration, annotators can be assigned to annotating and/or paraphrasing tasks. By clicking the assigned tasks, an annotator can access associated dialogues via the main annotating and paraphrasing interface.

3.2.1 Dialogue Annotating

As shown in Figure 2, the annotating interface is activated by a user's selection of the menu bar (①). The annotating interface consists of a left "usr" annotating zone (②), a middle utterance zone (③) and a right "sys" annotating zone (④). The following steps are performed to annotate a dialogue turn:

- **Utterance Editing**: Manual editing may be involved to fix typos or ASR-induced errors.

- **Domain and Intent Selection**: Both left and right annotating zones provide annotators with functions to select a general intent (⑤) of the current utterance, e.g., "Thank" or "Bye". The service domain (⑥) and all related intents (⑦) for the current utterance can be assigned in this zone.

- **Slot Annotating**: There are two ways to annotate a slot. A user can select a span of the utterance text (i.e., a restaurant name) and drag it to the slot-value editing box (⑧). The other way is to click the slot-value editing box and select one value from the opened drop-down list box. For an enumeration slot, only the second method is supported. For a string slot, the first method is used by default and whether the second method is available depends on the specific slot-values defined in the metadata.

During annotating, metaCAT enables automatical constraint-checking by validating the annotated results against predefined constraints (shown as ⑨ in Figure 2). The constraint-checking is activated under the circumstances depicted as below:

- **Turn Switching**: After an annotator finishes one dialogue turn, the validation is performed.

- **Validating or Saving**: When the "Validate" or "Save" button is clicked, metaCAT performs validating for all turns of dialogues.

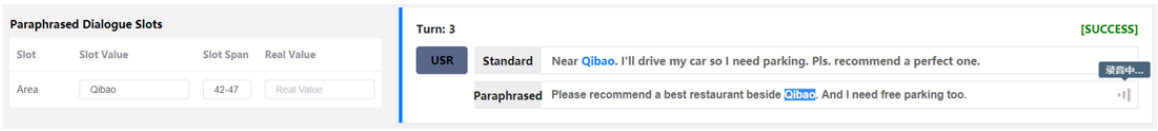

Figure 3: A screenshot of the paraphrasing interface of metaCAT.

Task		#Dialogue	#Turn/Dialogue	#Turn	#Non-span-slot	#Span-slot	#Slot
LIDA	Expert	79	3.5	276.5	816.5	0	816.5
	Novice	60	3.5	210	617	0	617
metaCAT	Expert	25	15.56	389	315.5	561.3	876.8
	Novice	20	15.56	311.2	252.4	449	701.4

Table 2: Hourly annotated dialogues in the re-annotation of 100 dialogues sampled from MultiWOZ 2.1 using metaCAT. "#" stands for "The number of".

Task	#Dialogue	#Turn	#Slot
ASR Exclusive	16.7	344.8	485.3
ASR Inclusive	28.6	612.4	863.7

Table 3: A comparison of paraphrasing and re-annotation of a bespoke cross-domain dialogue dataset with and without ASR during eight working hours. "#" stands for "The number of".

The validating process is designed to detect common annotating errors, e.g., repeated slots, repeated slot-values, overlapped spans, and ill-matched intents. The constraints are predefined or enforced by the metadata definition.

3.2.2 Dialogue Paraphrasing

The paraphrasing interface (Figure 3) is similar to the annotating interface with some minor differences. It is divided into two areas with the slot area on the left-hand side and the utterance area on the right side. The entire annotating operation includes the following steps:

- **Utterance Paraphrasing**: The annotator paraphrases the utterance using the colloquial language keeping the basic semantics, intent and slot unchanged.
- **Slot Annotating**: The annotator selects a span of the utterance and drags it to the slot-value editing box.
- **Real-value Filling**: For slots with real values, the annotator needs to fill the values to the corresponding slots after paraphrasing.

The validating process is basically the same as that of the annotating task except for the underlying detecting rules. There is no need to detect ill-matched intents, but the paraphrased utterance's integrity and diversity are checked quantitatively using Levenshtein distance.

4 Evaluation

metaCAT is used to correct annotation errors in MultiWOZ 2.1. One hundred dialogues are sampled from the test set and imported into metaCAT for evaluation. The benchmark data are those published in LIDA (Collins et al., 2019) on a different sample. Four annotators have been employed in this task, two of whom were novice users and the others are experienced annotators.

Table 2 captures the results of annotating a sample of MultiWOZ 2.1 using metaCAT on an hourly basis. Expert metaCAT users produce an average of 876.8 slot annotations, while novice users generate 701.4 annotations on average. Note that we cannot directly compare the results with those produced in LIDA (Collins et al., 2019) as different data are included in the experiment. However, the hourly annotation rate of metaCAT for both novice and expert users indicates an equivalent efficiency to those of LIDA. Table 3 shows that the inclusion of ASR increases the hourly annotation rate of paraphrasing and re-annotation by 1.8 folds.

LIDA is not able to handle metadata therefore dynamic domain switching is not available. Contrastively, It is easy to switch domain annotating as metaCAT clearly defines metadata and captures constraints among the domain, intent, slot and slot-value. In addition, drag-and-drop is an efficient way to handle span information annotation.

5 Conclusion and Future Work

We present metaCAT, an open-source web-based annotation tool designed specifically for task-oriented dialogue datasets. It is the first annotation tool providing a comprehensive metadata annotation to cover service domain, intent and span information. metaCAT ensures data quality with a real-time constraint-checking mechanism. An ASR-inclusive paraphrasing function enables users to generate more diversified utterances with annotations. metaCAT can be used to enhance task-oriented dialogue datasets (i.e., MultiWOZ 2.1) with easy-to-use functionality.

Existing dialogue annotation tools focus on the current turn of the dialogue, lacking inclusion of related contexts. For instance, there is a need to refer to the information from the previous turns to annotate the term "the hotel" in a typical hotel booking dialogue scenario. Apart from the above-mentioned co-referencing annotation function, future work will focus on designing other annotation features to meet the community's practical needs, for example, annotations for sentiment and multi-modal intent. Many open-domain conversations are goal-driven with the intent to solve real-world problems, for example, for recommendation and search purposes. Developing knowledge-grounded open-domain conversation datasets becomes an emerging trend (Gopalakrishnan et al., 2019) to regulate the free-form data with facts and knowledge. metaCAT can be customized to annotate open-domain dialogue datasets from this perspective.

Acknowledgments

We are grateful to colleagues from Huawei Technologies Co., Ltd. for insightful discussions and technical supports.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Edward Collins, Nikolai Rozanov, and Bingbing Zhang. 2019. LIDA: Lightweight interactive dialogue annotator. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 121–126, Hong Kong, China. Association for Computational Linguistics.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: an architecture for development of robust HLT applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 168–175, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyag Gao, and Dilek Hakkani-Tur. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tr. 2019. Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations. In *Proc. Interspeech 2019*, pages 1891–1895.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The INCEpTION platform: Machine-assisted and knowledge-oriented interactive annotation. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9, Santa Fe, New Mexico. Association for Computational Linguistics.

Brian Pluss. 2012. *TWIST Dialogue Annotation Tool*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset.

Martin Weisser. 2016. Dart the dialogue annotation and research tool. *Corpus Linguistics and Linguistic Theory*, 12(2):355 – 388.

Fan Yang and Peter A. Heeman. 2005. DialogueView: an annotation tool for dialogue. In *Proceedings of HLT/EMNLP 2005 Interactive Demonstrations*, pages 20–21, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Xiaoxue Zang, Abhinav Rastogi, and Jindong Chen. 2020. MultiWOZ 2.2 : A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117, Online. Association for Computational Linguistics.

Qi Zhu, Kaili Huang, Zheng Zhang, Xiaoyan Zhu, and Minlie Huang. 2020. CrossWOZ: A large-scale chinese cross-domain task-oriented dialogue dataset. *Transactions of the Association for Computational Linguistics*.

NLP Tools for Predictive Maintenance Records in MaintNet

Farhad Akhbardeh, Travis Desell, Marcos Zampieri
Rochester Institute of Technology, United States
{fa3019, tjdvse, mazgla}@rit.edu

Abstract

Processing maintenance logbook records is an important step in the development of predictive maintenance systems. Logbooks often include free text fields with domain specific terms, abbreviations, and non-standard spelling posing challenges to off-the-shelf NLP pipelines trained on standard contemporary corpora. Despite the importance of this data type, processing predictive maintenance data is still an under-explored topic in NLP. With the goal of providing more datasets and resources to the community, in this paper we present a number of new resources available in MaintNet, a collaborative open-source library and data repository of predictive maintenance language datasets. We describe novel annotated datasets from multiple domains such as aviation, automotive, and facility maintenance domains and new tools for segmentation, spell checking, POS tagging, clustering, and classification.

1 Introduction

Engineering systems are generating ever increasing amounts of maintenance records often recorded in the form of event logbooks. The analysis of these records are aimed to improve predictive maintenance systems reducing maintenance costs, helping to prevent accidents, and saving lives (Jarry et al., 2018). Predictive maintenance records are collected in multiple domains such as aviation, healthcare, and transportation (Tanguy et al., 2016; Altuncu et al., 2018). In this paper, we present new datasets in the aviation and automotive domains listed in Table 2.

Maintenance record datasets generally contain free text fields describing issues and actions, as in the instances presented in Table 1. Most standard NLP pipelines for pre-processing and annotation are trained on standard contemporary corpora (e.g. newspaper texts, novels) failing to address most of the domain specific terminology, abbreviations, and non-standard spelling present in maintenance records. To help support research in this area, the MaintNet[1] platform, a collaborative open-source library and data repository for predictive maintenance data, has been developed (Akhbardeh et al., 2020). In this paper, we present an evaluation of the tools available at MaintNet, as well as two new datasets included in the platform.

The main contributions of this paper are the following:

1. The creation of novel language resources (e.g. abbreviation lists, datasets, and termbanks) for technical language and predictive maintenance data in the aviation, automotive, and facility management domains. We present two new datasets with aviation and automotive safety records that have been recently collected and annotated and are now available at MaintNet.

2. The creation and development of manually curated gold standards that can be used to evaluate the performance of POS tagging and clustering/classification on technical logbook data.

3. The development and evaluation of a number of Python (pre-)processing tools available at MaintNet including stop word removal, stemmers, lemmatizers, POS tagging, and clustering. We carry out an evaluation of MaintNet's spell checkers and POS taggers comparing them to off-the-shelf NLP packages such as NLTK (Bird et al., 2009) and Stanford Core NLP (Manning et al., 2014), as well as clustering methods.

[1] Available at: https://people.rit.edu/fa3019/MaintNet/

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 26–32

ID	Issue/Problem	Date	Action
111552	R/H FWD UPPER BAFL SEAL NEEDS TO BE RESECURED	7/2/2012	INSTALLED POP RIVET TO RESECURE R/H FWD BAF SEEAL.
111563	CAP SCREWE MISSING, L/H ENG #4 BAFLE	7/3/2012	INSTALLED NEW SCREW. CHKD ENG
111574	CYL #1 BAFFLE CRACKED AT SCREW SUPPORT & FWD BAFL BELOWE #1	7/2/2012	FABRICATED PATCHES OF LIKE MATERIAL & RIVETED IAW CESSN
111585	#3 FWD PUSH ROD TUBE GSK LEAKING @ EGNINE	7/2/2012	REMOVED & REPLACED #3 FWD PUSH ROD TUBE SEALS. LEAK CHE

Table 1: Four instances from one of MaintNet's aviation datasets.

Domain	Dataset	Inst.	Tokens	Code	Source
Aviation	Maintenance	6,169	76,866	*Avi-Main*	University of North Dakota Aviation Program
	Accident	5,268	162,533	*Avi-Acc*	Open Data by Socrata
	Safety	**25,558**	345,979	***Avi-Safe***	**Federal Aviation Administration**
Automotive	Maintenance	617	4,443	*Auto-Main*	Connecticut Open Data
	Accident	54,367	242,012	*Auto-Acc*	NYS Department of Motor Vehicles
	Safety	**5,456**	**137,038**	***Auto-Safe***	**Open Data DC**
Facility	Maintenance	87,276	2,469,003	*Faci-Main*	Baltimore City Maryland Preventive Maintenance

Table 2: Instances and tokens in each dataset in MaintNet. Two new datasets, (*Avi-safe* and *Auto-Safe*), displayed in bold.

2 Related Work

Research in predictive maintenance systems requires large, cleansed, and often annotated logbook data gathered in domains such as web information extraction, system maintenance (*e.g.*, aviation, wind turbines, automobiles), and healthcare (*e.g.*electronic health records).

In the domain of healthcare, Altuncu et al. (2018) analyzed health records of patient incidents provided by the UK National Health Service using a deep neural network with word embedding. Tixier et al. (2016) developed a system to analyze injury reports applying POS tagging and term frequency to extract keywords about injuries creating a dictionary of events to improve future safety management. Savova et al. (2010) applied off-the-shelf NLTK libraries on free-text electronic medical records for information extraction purposes.

In technical domains such as aviation, where MaintNet provides a primary resource, Tanguy et al. (2016) studied various available NLP techniques such as topic modeling to process aviation incident reports and extract useful information. They used standard NLP libraries to pre-process the data and then applied the Talismane NLP toolkit (Urieli, 2013) for incident feature extraction and training.

As to the problem of non-standard spelling, Siklósi et al. (2013), proposed a method of correcting misspelled words in clinical records by mapping spelling errors to a large database of correction candidates. However, due to the large number of abbreviations in medical records, they were limited to specific terms and the normalization had to be performed separately. de Amorim and Zampieri (2013) proposed a dictionary-based spell correction algorithm using a clustering technique by comparing various distance metrics to aim to lower the number of distance calculations while finding or matching target words for misspellings. With this in mind, in MaintNet we provide users with tools developed to deal with domain-specific misspellings and abbreviations.

3 MaintNet Features

In the next sub-sections we present the tools in resources available in MaintNet divided into language resources, pre-processing, and clustering, In addition to that, MaintNet provides various dynamic webpages for users to communicate with each other and with the project developers which work similarly to a forum or message board. We hope that MaintNet's community participation features will further facilitate discussion and research in this under explored domain.

3.1 Language Resources

MaintNet currently features seven English datasets from the aviation, automotive, and facility maintenance domains, which are presented in Table 2. This paper introduces two new datasets with aviation and automotive safety records in bold. These datasets were collected from the USA Federal Aviation Administration and Open Data DC respectively. The list of fields and data types in each dataset is presented in Table 3.

Code	Fields and Data Types
Avi-Main	problem, action (text), ata chapter code (int.), date opened/closed (date), identifier/work order (int.)
Avi-Acc	flight (date), type (text), summary of incident (text), record/flight (int.)
Avi-Safe	flight (date), indicated safety/damage type (text), safety remarks (text), flight phase (text), identifier number (int.)
Auto-Main	problem, action (text), reason (text), department (int.), date opened/closed (date), identifier/job number (int.)
Auto-Acc	reported accident (text), accident or injury type (text), date issued (date), record number (int.)
Auto-Safe	request type (text), comment/report (text), request identifier number (int.)
Faci-Main	problem, action, problem type (text), location (text), date opened/closed (date), identifier/Work number (int.)

Table 3: Fields and data types in MaintNet's datasets.

In Figure 1 we present a screenshot of one of MaintNet's datasets, the *Avi-Main* dataset, that can be accessed and searched through the platform. Predictive maintenance datasets are particularly hard to obtain due to the sensitive information they contain. Therefore, we work closely with the data providers to ensure that all confidential and sensitive information in all datasets remains anonymous. As a collaborative platform, MaintNet will be expanded with the collaboration from interested members of the NLP community.

MaintNet further provides the user with domain specific abbreviation dictionaries, morphosyntactic annotation, and term banks validated by domain experts. The morphosyntactic annotation contains the POS tag, compound, lemma, and word stems. Finally, the domain term banks contain a list of terms that are used in each domain along with a sample of usage extracted from the corpus.

3.2 Pre-processing and Tools

One of the bottlenecks of automatically processing logbooks for predictive maintenance system is that most of these datasets are not annotated with the reason for maintenance or a categorization of the issue type. To address this issue, we implemented several pre-processing steps to clean and extract as much information from logbooks as possible. The pipeline is shown in Figure 2. The Python scripts for all components in this pipeline are made available through Maintnet.

The process starts with text normalization, including lowercasing, stop word and punctuation removal, and treating special characters with NLTK's (Bird et al., 2009) regular expression library, followed by tokenization (NLTK tokenizer), stemming (Snowball Stemmer), and lemmatization (WordNet (Miller, 1992)). With use of the collected morphosyntactic information, POS annotation is carried out with the NLTK POS tagger. Term frequency-inverse document frequency (TF-IDF) is obtained using the *gensim tfidf model* (Rehurek and Sojka, 2010). Our analysis of the logbooks found that many of the misspellings and abbreviations lead to incorrect or non-existent dictionary look ups. To overcome this issue, we explored various state-of-the-art spellcheckers including Enchant[2], Pyspellchecker[3], Symspellpy[4], and Autocorrect[5].

Given the inaccuracy of existing techniques, we developed methods of correcting syntactic errors, typos, and abbreviated words using a Levenshtein (Levenshtein, 1966). This method uses a dictionary of domain specific words and maps the various possible misspelled words into the correct format by selecting the most similar word in the dictionary. The Levenshtein algorithm was chosen over other distance metrics (*e.g.*, Euclidian, Cosine) as it allows us to control the minimum number of string edits and its widely used in spell checking (de Amorim and Zampieri, 2013). The results of our method compared to other spellchecking techniques in random samples of 500 instances from each of the 5 datasets is presented in Table 4.

The results are reported in terms of success rate showing that the Levenshtein (Lev) algorithm

[2] https://www.abisource.com/projects/enchant/

[3] https://github.com/barrust/pyspellchecker

[4] https://github.com/wolfgarbe/SymSpell

[5] https://github.com/fsondej/autocorrect

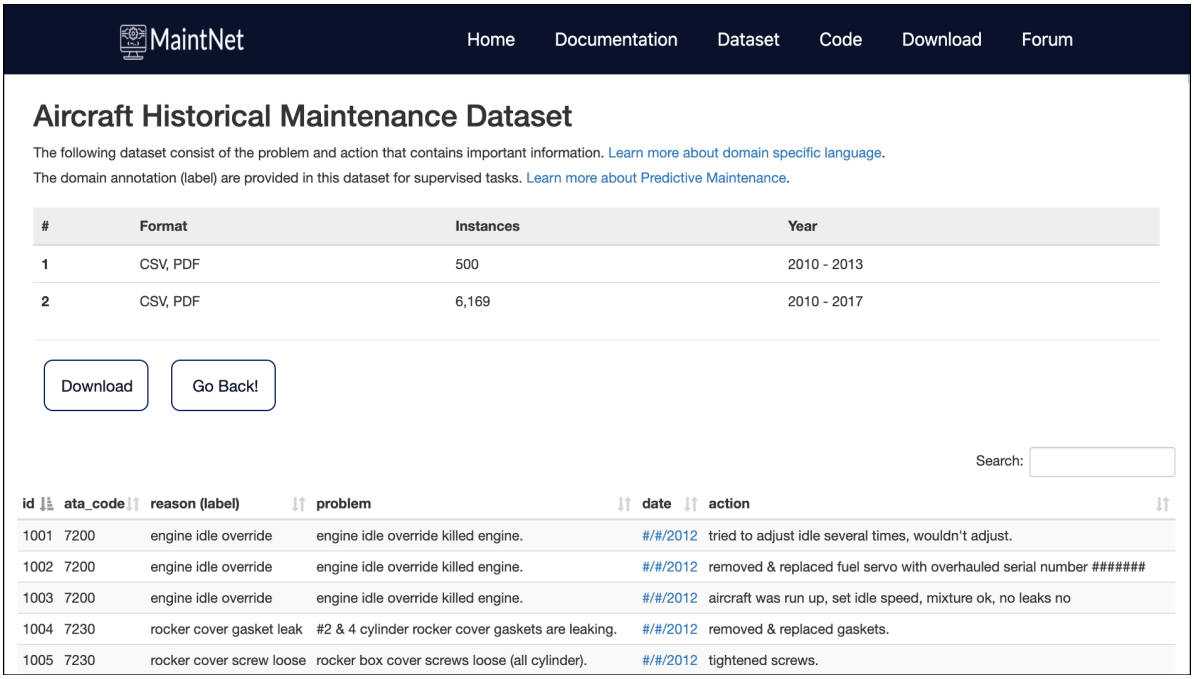

Figure 1: A screenshot of one of MaintNet's dataset webpages.

outperforms the Enchant (Ench), Pyspellchecker (Spell), and Autocorrect (Auto) spell checkers.

Code	Token	Miss	Ench	Spell	Auto	Lev.
Avi-Main	3299	289	86%	61%	73%	98%
Avi-Safe	6059	828	84%	56%	68%	91%
Auto-Main	2599	266	69%	27%	49%	95%
Auto-Acc	2422	169	87%	59%	77%	97%
Faci-Main	7758	926	83%	63%	59%	93%

Table 4: Success rate of spell checkers on 500 instances per dataset. Token stands for total tokens and Miss stands for misspelled tokens.

WordNet was used to lemmatize the document, however it requires defining a POS tagger parameter which we want to lemmatize (the wordNet default is "noun"). As the maintenance instances typically consist of verb, noun, adverb and adjective words that define a problem, action and occurrence, by using "verb" as the POS parameter, there is an issue of mapping important noun words such as "left" (*e.g.* left engine) to "leave" or "ground" to "grind". To resolve this issue, as we discussed in 3.1, we created an exception list using developed morphosyntactic information for the WordNet lemmatizer to ignore mapping words which could be multiple parts of speech.

Finally, we have performed an extrinsic evaluation of MaintNet's pre-processing pipeline by evaluating its impact on POS tagging. To carry out this evaluation, we randomly selected 500 instances of the *Avi-Main* dataset to serve as our gold standard. A North-American English native speaker working in the project annotated the 500 instances using the Penn Treebank tagset. We make this gold standard available to the community in MaintNet.

We compared the performance of three available POS taggers: NLTK (Bird et al., 2009), Stanford CoreNLP (Manning et al., 2014) and TextBlob[6] trained on the raw and pre-processed versions the *Avi-Main* dataset and evaluated on raw and pre-processed versions of the gold standard. We present the results in Table 5 in terms of accuracy. Stanford CoreNLP obtained the best results among the three POS taggers with 91% and 87% accuracy on the processed and raw versions of the data respectively. The results show an improvement of 4% accuracy in the performance of each of the three POS taggers when annotating MaintNet's pre-processed data confirming the importance of these pre-processing methods.

POS Tagger	Raw	Processed	Difference
NLTK	77%	81%	+4%
Stanford	87%	91%	+4%
TextBlob	77%	81%	+4%

Table 5: Results of three POS taggers annotating raw and (pre-)processed versions of the gold standard.

[6] https://textblob.readthedocs.io/en/dev/

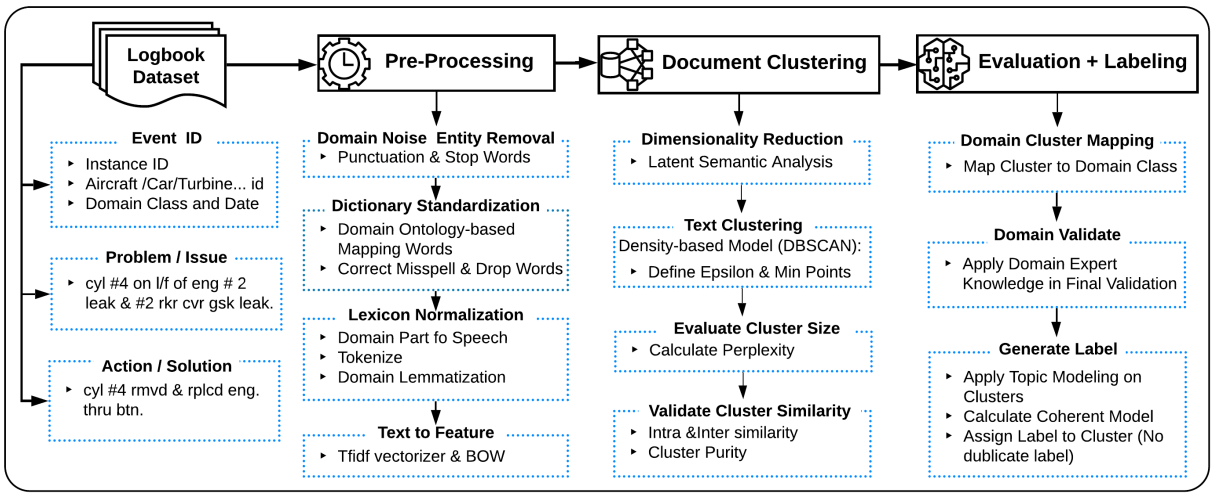

Figure 2: The components in MaintNet's processing and information extraction pipeline: pre-processing, document clustering, and evaluation.

3.3 Clustering

MaintNet also features implementations of popular clustering algorithms applied to logbook data that are made freely available to the research community. The motivation behind this is that most logbook data available is not annotated, which requires a domain expert to group instances into categories. Clustering techniques were used to help in this process.

We converted the terms and words into a numerical representation using libraries such as *tfidfvectorizer* (ElSahar et al., 2017) resulting in a large matrix of document terms (DT). We use truncated singular value decomposition (SVD) (ElSahar et al., 2017) known as latent semantic analysis (LSA), to perform a linear dimensionality reduction. We chose truncated SVD (LSA) over principal component analysis (PCA) (ElSahar et al., 2017) in our system, due to the fact LSA can directly be applied to our *tfidf* DT matrix and it focuses on document and term relationships where PCA focuses on a term covariance matrix (eigendecomposition of the correlation). We experimented with different 4 clustering techniques: k-means (Jain, 2010), Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996), Latent Dirichlet Analysis (LDA) (Vorontsov et al., 2015), and hierarchical clustering (Aggarwal and Zhai, 2012). For comparison of the results, the silhouette and inertia metrics (Fraley and Raftery, 1998) were used to determine the number of clusters for k-means (both provided similar results), and perplexity (Fraley and Raftery, 1998) and coherence (Vorontsov et al., 2015) scores were used for LDA. DBSCAN and hierarchical clustering do not require a predetermined number of clusters.

For evaluation, we used a standard measurement of cluster cohesion including high intra-cluster similarity and low inter-cluster similarity. We chose 3 different similarity algorithms including Levenshtein, Jaro, and cosine (Fraley and Raftery, 1998) to calculate intra- and inter-cluster similarity. The cosine similarity metric is commonly used and is independent of the length of document, while Jaro is more flexible by providing a rating of matching strings. We collected human annotated instances by a domain expert to serve as our gold standard, and these are provided on MaintNet to encourage research into improving unsupervised clustering of maintenance logbooks.

Finally, Figure 3 shows the empirical analysis of the four clustering techniques with and without our additional data pre-processing steps (Levenshtein-based dictionary spellchecking and the lemmatizer list previously presented) on the *Avi-Main* dataset. We examined the distribution of cluster sizes, the number of clusters, and the number of outliers (in the case of DBSCAN). Using a domain-based spellchecker and the modified lemmatizer list improved the purity and overall accuracy of the clusters by increasing the means of intra-cluster similarity and decreasing the means of inter-cluster similarity.

The DBSCAN provided more accurate clusters in comparison to other algorithms while also detecting outliers, which could help identify if any new issues are introduced to the maintenance logs or if there are safety issues reported by the pilot

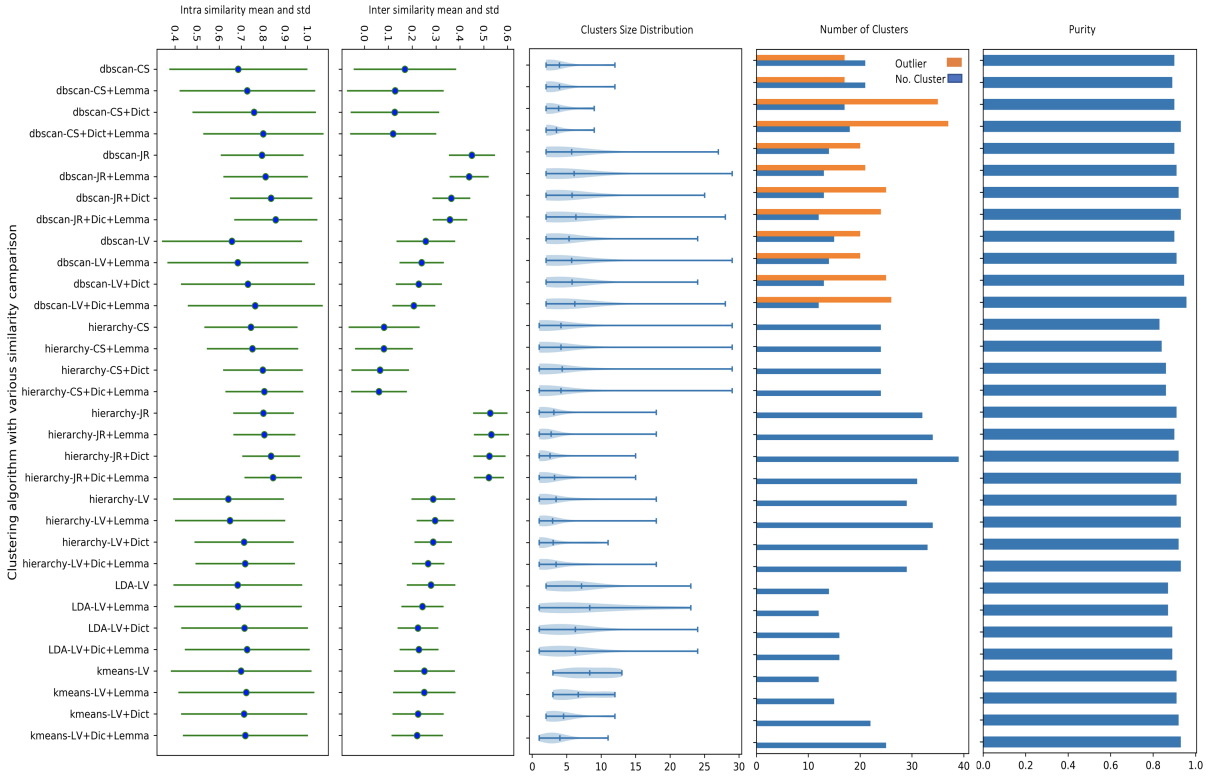

Figure 3: Results of the clustering methods. From left to right, calculated mean and standard deviation of intra- and inter-cluster similarity, cluster size distribution, number of clusters generated by each method and purity on *Avi-Main* dataset.

during flight operation. K-means provided somewhat comparable results to DBSCAN, but it was not able to detect outliers and determining the number of clusters (K) is challenging, especially as this number may change over time as more issues are reported. Hierarchical clustering performed poorly, where similar issues were found to be distributed across different clusters. It was also more computationally expensive than the other methods. Clusters generated with LDA were better than hierarchical clustering, however LDA clustered some of the documents that contain the same equipment with different types of issues description together, resulting in clusters with a mixture of issue types.

4 Conclusion

In this paper we evaluate the tools available in MaintNet, a collaborative open-source library for predictive maintenance language resources. MaintNet provides technical logbook datasets on multiple domains: aviation, automotive, and facility maintenance. A number of other important language resources such as abbreviation lists, morphosyntactic information lists, and termbanks have been developed and are also available through the platform. Text (pre-)processing tools developed in Python were evaluated and are also made available. These include spell checking, POS tagging, and document clustering.

Finally, we performed an intrinsic evaluation comparing the performance of several spell-checkers on five of the seven datasets in MaintNet and an extrinsic evaluation on raw and processed versions of the *Avi-Main* dataset on POS tagging. We showed an important increase in performance for all taggers we tested when using data processed with MaintNet's pre-processing pipeline. For the POS tagger comparison and clustering, we developed manually annotated gold standards which are also made available through the platform.

Acknowledgments

We thank the University of North Dakota's aviation program for providing the aviation maintenance records dataset.

We also thank Zechariah Morgain, the aviation domain expert who evaluated the results of our pre-processing techniques and clustering algorithms at many stages during this work, providing us with detailed information about these data sets.

References

Charu C. Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. In *Mining Text Data*.

Farhad Akhbardeh, Travis Desell, and Marcos Zampieri. 2020. MaintNet: A Collaborative Open-Source Library for Predictive Maintenance Language Resources. *arXiv preprint arXiv:2005.12443*.

M. Tarik Altuncu, Erik Mayer, Sophia N. Yaliraki, and Mauricio Barahona. 2018. From text to topics in healthcare records: An unsupervised graph partitioning methodology. *ArXiv*, abs/1807.02599.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly.

Renato Cordeiro de Amorim and Marcos Zampieri. 2013. Effective spell checking methods using clustering algorithms. In *RANLP*.

Hady ElSahar, Elena Demidova, Simon Gottschalk, Christophe Gravier, and Frédérique Laforest. 2017. Unsupervised open relation extraction. *ArXiv*, abs/1801.07174.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*.

Chris Fraley and Adrian E. Raftery. 1998. How many clusters? which clustering method? answers via model-based cluster analysis. *Comput. J.*, 41:578–588.

Anil Kumar Jain. 2010. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31:651–666.

Gabriel Jarry, Daniel Delahaye, Florence Nicol, and Eric Feron. 2018. Aircraft atypical approach detection using functional principal component analysis. In *SID*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL*.

George A. Miller. 1992. Wordnet: A lexical database for english. *Commun. ACM*, 38:39–41.

Radim Rehurek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *LREC*.

Guergana K. Savova, James J. Masanz, Philip V. Ogren, Jiaping Zheng, Sunghwan Sohn, Karin Kipper Schuler, and Christopher G. Chute. 2010. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association : JAMIA*, 17 5:507–13.

Borbála Siklósi, Attila Novák, and Gábor Prószéky. 2013. Context-aware correction of spelling errors in hungarian medical documents. In *SLSP*.

Ludovic Tanguy, Nikola Tulechki, Assaf Urieli, Eric Hermann, and Céline Raynal. 2016. Natural language processing for aviation safety reports: From classification to interactive analysis. *Computers in Industry*, 78:80–95.

Antoine J.-P. Tixier, Matthew R. Hallowell, Balaji Rajagopalan, and Dean Bowman. 2016. Automated content analysis for construction safety: A natural language processing system to extract precursors and outcomes from unstructured injury reports. In *Automation in Construction*.

Assaf Urieli. 2013. *Robust French Syntax Analysis: Reconciling Statistical Methods and Linguistic Knowledge in the Talismane Toolkit*. Ph.D. thesis, Université de Toulouse II le Mirail.

Konstantin Vorontsov, Oleksandr Frei, Murat Apishev, Peter Romov, and Marina Dudarenko. 2015. Bigartm: Open source library for regularized multimodal topic modeling of large collections. In *AIST*.

FAIRSEQ S2T: Fast Speech-to-Text Modeling with FAIRSEQ

Changhan Wang[1], **Yun Tang**[1], **Xutai Ma**[1,2], **Anne Wu**[1], **Dmytro Okhonko**[1], **Juan Pino**[1]

[1]Facebook AI
[2]Johns Hopkins University

xutai_ma@jhu.edu
{changhan,yuntang,xutaima,annewu,oxo,juancarabina}@fb.com

Abstract

We introduce FAIRSEQ S2T, a FAIRSEQ (Ott et al., 2019) extension for speech-to-text (S2T) modeling tasks such as end-to-end speech recognition and speech-to-text translation. It follows FAIRSEQ's careful design for scalability and extensibility. We provide end-to-end workflows from data pre-processing, model training to offline (online) inference. We implement state-of-the-art RNN-based as well as Transformer-based models and open-source detailed training recipes. FAIRSEQ's machine translation models and language models can be seamlessly integrated into S2T workflows for multi-task learning or transfer learning. FAIRSEQ S2T documentation and examples are available at https://github.com/pytorch/fairseq/tree/master/examples/speech_to_text.

1 Introduction

End-to-end sequence-to-sequence (S2S) modeling has witnessed rapidly increased applications in speech-to-text (S2T) tasks. It achieves state-of-the-art performance on automatic speech recognition (ASR) (Park et al., 2019; Synnaeve et al., 2019) and leads to the recent resurgence of speech-to-text translation (ST) research (Duong et al., 2016; Bérard et al., 2016). ASR and ST are closely related. There are recent attempts to combine the two tasks under the same S2S model architecture via multi-task learning (Anastasopoulos and Chiang, 2018; Liu et al., 2020). They also benefit from each other via transfer learning (Bansal et al., 2019; Wang et al., 2020b) and are able to leverage additional supervision from machine translation (MT) and language modeling (LM). When supervised data is not abundant, self-supervised pre-training (Schneider et al., 2019; Wu et al., 2020) and semi-supervised training (Kahn et al., 2020; Pino et al., 2020) lowers the requirements on supervision and improves model performance.

The increased connections among ASR, ST, MT and LM has called for all-in-one S2S modeling toolkits, and the use of large-scale unlabeled speech data sets the scalability requirements. In this paper, we introduce FAIRSEQ S2T, a FAIRSEQ (Ott et al., 2019) extension for S2T tasks such as end-to-end ASR and ST. It follows FAIRSEQ's careful design for scalability and extensibility. We provide end-to-end workflows from data pre-processing, model training to offline (online) inference. We implement state-of-the-art RNN-based (Chan et al., 2016; Bérard et al., 2018) and Transformer-based (Vaswani et al., 2017; Mohamed et al., 2019) models and open-source detailed training recipes. FAIRSEQ's MT models and LMs can be seamlessly integrated into S2T workflows for multi-task learning or transfer learning. To facilitate model evaluation, we add a collection of scorers as well as VizSeq (Wang et al., 2019) integration for visualized error analysis. FAIRSEQ S2T documentation and examples are available at https://github.com/pytorch/fairseq/tree/master/examples/speech_to_text.

With counterpart toolkits such as ESPNet (Inaguma et al., 2020) and Lingvo (Shen et al., 2019), FAIRSEQ S2T pursues the best integration, scalability and reproducibility. A detailed comparison of FAIRSEQ S2T with its counterparts can be found in Table 1.

2 Features

Fairseq Models FAIRSEQ provides a collection of MT models (Ng et al., 2019; Lewis et al., 2020) and LMs (Liu et al., 2019; Conneau et al., 2020) that demonstrate state-of-the-art performance on standard benchmarks. They are open-sourced with pre-trained models. FAIRSEQ also supports other tasks such as text summarization, story generation and self-supervised speech pre-training.

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 33–39

	ASR	LM	MT	Non-Autoreg. MT	Offline ST	Online ST	Speech Pre-training	Multi-node training	Pre-trained models
ESPNet-ST	✓	✓	✓		✓			✓†	✓
Lingvo	✓	✓	✓		✓‡			✓	
OpenSeq2seq[1]	✓	✓	✓					✓	✓
RETURNN[2]	✓	✓	✓		✓			✓	✓
SLT.KIT[3]	✓		✓		✓				✓
Tensor2Tensor[4]	✓	✓	✓					✓	✓
OpenNMT[5]	✓	✓	✓					✓	✓
Kaldi[6]	✓	✓							✓
Wav2letter++[7]	✓	✓							✓
fairseq S2T	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of FAIRSEQ S2T with counterpart toolkits (as of July 2020). † Only available in version 2 (under development). ‡ Not publicly available. [1] Kuchaiev et al. (2018). [2] Zeyer et al. (2018). [3] Zenkel et al. (2018). [4] Vaswani et al. (2018). [5] Klein et al. (2017). [6] Povey et al. (2011). [7] Pratap et al. (2018).

S2T extension FAIRSEQ S2T adds attention-based RNN models (Chan et al., 2016; Bérard et al., 2018) as well as the latest Transformer models (Vaswani et al., 2017; Mohamed et al., 2019) for ASR and ST. It also supports CTC criterion (Graves et al., 2006) for ASR. For the simultaneous ST setting, it includes online models with widely used policies: monotonic attention (Raffel et al., 2017), wait-k (Ma et al., 2019), monotonic infinite look-back attention (Arivazhagan et al., 2019b), and monotonic multihead attention (Ma et al., 2020b).

Data Pre-Processing FAIRSEQ S2T extracts Kaldi-compliant (Povey et al., 2011) speech features (e.g. log mel-filter banks) automatically from WAV/FLAC audio files via PyKaldi (Can et al., 2018) or torchaudio[1]. Speech features can also be pre-computed and stored in NumPy (Harris et al., 2020) format. Optionally, raw audio files or features files can be packed into ZIP archives to improve I/O performance or facilitate file management. For further pre-processing, FAIRSEQ S2T provides online speech data transforms, including CMVN (cepstral mean and variance normalization), speed perturbation (Ko et al., 2017) and SpecAugment (Park et al., 2019). It also has an open interface for user-defined transforms. For text data, FAIRSEQ S2T does online tokenization with a rich collection of tokenizers, including Moses[2], SentencePiece (Kudo and Richardson, 2018), subword-nmt[3], byte-level BPE (Wang et al., 2020a) and bytes (Li et al., 2019).

Data Configuration FAIRSEQ S2T gets raw audio (feature) paths and target texts from manifest files in TSV (tab-separated values) format, which is similar to Kaldi-style scp files. Online speech data transforms and other data-related settings (e.g. tokenizer type and vocabulary) are defined by a separate configuration file in YAML format.

Computation FAIRSEQ is implemented in PyTorch (Paszke et al., 2019) and it provides efficient batching, mixed precision training (Micikevicius et al., 2018), multi-GPU as well as multi-machine training for computational efficiency on large-scale experiments.

Evaluation Metrics FAIRSEQ S2T provides common automatic metrics for ASR, ST and MT, including WER (word error rate), BLEU (Papineni et al., 2002) and chrF (Popović, 2015). It also integrates SIMULEVAL (Ma et al., 2020a) for simultaneous ST/MT metrics such as AL (average lagging) (Ma et al., 2019) and DAL (differentiable average Lagging) (Cherry and Foster, 2019).

Visualization FAIRSEQ supports Tensorboard[4] for monitoring holistic metrics during model training. It also has VizSeq (Wang et al., 2019) integration for sequence-level error analysis, where speech and target/predicted text data are visualized with alignments in Jupyter Notebook interface.

3 Experiments

We evaluate FAIRSEQ S2T models on English ASR benchmark—LibriSpeech (Panayotov et al., 2015), as well as multilingual ST benchmarks—MuST-C (Di Gangi et al., 2019a) and CoVoST 2 (Wang

[1] https://github.com/pytorch/audio
[2] https://github.com/moses-smt/mosesdecoder
[3] https://github.com/rsennrich/subword-nmt
[4] https://github.com/tensorflow/tensorboard

		De	Nl	Es	Fr	It	Pt	Ro	Ru
	Transformer[1]	17.3	18.8	20.8	26.9	16.8	20.1	16.5	10.5
	Transformer[2†]	22.9	27.4	28.0	32.7	23.8	28.0	21.9	15.8
	T-Sm	22.7	27.3	27.2	32.9	22.7	28.1	21.9	15.3
	Multi. T-Md[*]	24.5	28.6	28.2	34.9	24.6	31.1	23.8	16.0
B-Base	Offline	19.2	23.5	24.0	29.1	16.4	23.5	19.7	13.7
B-Base	High Lat.[‡]	18.6 (6.8)	22.9 (6.9)	22.3 (6.8)	28.4 (6.7)	15.4 (6.8)	22.6 (6.9)	19.1 (6.7)	12.9 (6.9)
B-Base	Mid Lat.[‡]	14.1 (5.4)	17.9 (5.4)	17.2 (5.5)	25.0 (5.3)	12.0 (5.5)	17.7 (5.8)	15.0 (5.6)	7.2 (5.8)
B-Base	Low Lat.[‡]	8.2 (2.9)	12.3 (2.8)	13.0 (3.0)	21.1 (2.8)	6.7 (2.9)	13.3 (2.9)	12.1 (2.9)	4.9 (2.7)

Table 2: FAIRSEQ S2T models on MuST-C. Test BLEU reported (for online models, AL is shown in parentheses). [1] Di Gangi et al. (2019). [2] Inaguma et al. (2020). [†] Applied additional techniques: speed perturbation, pre-trained decoder from MT and auxiliary CTC loss for ASR pre-training. [‡] Online models using beam size of 1 (instead of 5). [*] Trained jointly on all 8 languages.

	Type	Config.	Params
B-Base	RNN[†]	512d, 3L enc./2L dec.	31M
B-Big		512d, 5L enc./3L dec.	52M
T-Sm	Trans-	256d, 12L enc./6L dec.	31M
T-Md	form-	512d, 12L enc./6L dec.	72M
T-Lg	er[‡]	1024d, 12L enc./6L dec.	263M

Table 3: FAIRSEQ S2T models for benchmarking. For simplicity, we use the same (default) model hyper-parameters and learning rate schedule across all experiments. [†] Bérard et al. (2018). [‡] Vaswani et al. (2017).

	Dev		Test	
	Clean	Other	Clean	Other
LAS[†]	-	-	2.8	6.8
Transformer[‡]	2.5	6.7	2.9	7.0
B-Big	3.7	11.4	3.9	11.5
T-Sm	4.1	9.3	4.4	9.2
T-Md	3.5	8.1	3.7	8.1
T-Lg	3.3	7.7	3.5	7.8

Table 4: FAIRSEQ S2T models on LibriSpeech (using default hyper-parameters and LR schedule). Dev and test WER reported. [†] Park et al. (2019). [‡] Synnaeve et al. (2019).

et al., 2020c). The model architectures used in benchmarking can be found in Table 3.

3.1 Experimental Setup

For speech inputs, we extract 80-channel log mel-filter bank features (25ms window size and 10ms shift) with utterance-level CMVN applied. We remove training samples with more than 3,000 frames for GPU memory efficiency. To alleviate overfitting, we pre-train ST model encoders on English ASR and adopt SpecAugment (without time warping): LD policy on LibriSpeech models and LB policy on MuST-C and CoVoST 2 models. We average the last 10 checkpoints and use a beam size of 5 for decoding. For ASR, we use 10K unigram vocabulary (Kudo and Richardson, 2018) and report WER. For ST, we use character vocabulary for CoVoST 2 and 8K unigram vocabulary for MuST-C. We report case-sensitive detokenized BLEU using sacreBLEU (Post, 2018), except for Japanese and Chinese translations (no word segmentation) where we report character-level BLEU.

3.2 Speech Recognition (ASR)

LibriSpeech is a de-facto standard ASR benchmark that contains 1,000 hours of English speech from audiobooks. Table 4 shows the dev and test WER of our models on LibriSpeech clean and noisy sets. Two popular architectures, RNN-based model ("B-Big") and Transformer based models ("T-Sm", "T-Md" and "T-Lg"), are evaluated. We can see that both architectures are able to achieve competitive performance (WER) to the state-of-the-art ones (the upper section), while we use only default model hyper-parameters and learning rate schedule without any task-specific tuning.

3.3 Speech Translation (ST)

3.3.1 MuST-C

MuST-C contains up to around 500 hours of English speech from TED talks with translations in 8 European languages. Table 2 shows the test BLEU of our Transformer-based models ("T-Sm" and "Multi. T-Md") and RNN-based models ("B-Base") on all the MuST-C language directions. Compared with previous Transformer-based approaches (Di Gangi et al., 2019b; Inaguma et al., 2020), our bilingual models achieve comparative results to the state of the art without applying additional techniques such as speed perturbation and

	Fr	De	Es	Zh	Tr	Ar	Sv	Lv	Sl	Ta	Ja	Id	Cy
						X→En							
B-Base	23.2	15.7	20.2	4.4	2.2	2.7	1.4	1.2	1.5	0.2	1.1	1.0	1.7
+ SSL*	23.1	16.2	20.2	4.8	3.2	3.8	3.7	2.3	2.2	0.2	1.6	1.6	2.2
Multi. B-Big†	26.6	19.5	26.3	4.4	2.1	0.3	1.3	0.6	1.4	0.1	0.6	0.3	0.9
T-Sm	26.3	17.1	23.0	5.8	3.6	4.3	2.7	2.5	3.0	0.3	1.5	2.5	2.7
Multi. T-Md†	26.5	17.5	27.0	5.9	2.3	0.4	0.5	0.6	0.7	0.1	0.1	0.3	1.9
						En→X							
B-Base	-	12.5	-	20.0	6.7	9.1	18.1	8.7	11.6	7.4	25.6	15.2	18.9
Multi. B-Big‡	-	12.6	-	22.2	7.3	8.0	18.3	8.9	11.4	7.3	28.2	16.0	19.3
T-Sm	-	16.3	-	25.4	10.0	12.1	21.8	13.0	16.0	10.9	29.6	20.4	23.9
Multi. T-Md‡	-	15.4	-	26.5	9.5	10.8	20.9	12.2	14.6	10.3	30.5	18.9	22.0

Table 5: FAIRSEQ S2T models on CoVoST 2. Test BLEU reported (character-level BLEU for Zh and Ja targets). * Replaced mel-filter bank features with wav2vec ones (Schneider et al., 2019; Wu et al., 2020). † Trained jointly on all 21 X-En directions with temperature-based (T=2) resampling (Arivazhagan et al., 2019a). ‡ Trained jointly on all 15 En-X directions.

pre-trained decoder from MT. Moreover, our multilingual model (trained on all 8 languages) outperforms all bilingual ones with large margins. Besides traditional offline models, we also provide simultaneous ST models: the lower section in Table 2 presents the online models with wait-k policy, which was the baseline system in the IWSLT 2020 shared task on simultaneous ST (Ansari et al., 2020). The results represent the best systems in high (AL $>$ 6), medium (6 $\geq$ AL $>$ 3) and low (AL $\leq$ 3) latency regimes, on which we can clearly see the trade-offs between model performance and prediction latency.

3.3.2 CoVoST 2

CoVoST 2 contains total 2,880 hours of read speech in 22 languages from the open-source community, with 21 X-En directions and 15 En-X directions. We evaluate our models bidirectionally on 13 languages of them, including low-resource X-En directions: Zh, Tr, Ar, Sv, Lv, Sl, Ta, Ja, Id and Cy. We observe from Table 5 that our Transformer-based models ("T-Sm" and "T-Md") outperforms RNN-based ones ("B-Base" and "B-Big") on all En-X and X-En directions. The performance gap tends to be larger when the training data is higher resource (En-X directions, Fr-En, De-En and Es-En). Our multilingual models perform reasonably well with a universal model for over 15 X-En or En-X directions. They even have significant improvements on some directions (e.g. at least 4 BLEU gain on Es-En). For low-resource directions, we also evaluate self-supervised speech features (Schneider et al., 2019; Wu et al., 2020)[5] as an alternative to the traditional log mel-filter bank features ("+ SSL"). We find that self-supervised features bring consistent gains and transfer well across different languages (self-supervised model trained on English and feature extracted for non-English).

4 Conclusion

We introduce FAIRSEQ S2T, a FAIRSEQ extension for speech-to-text (S2T) modeling tasks such as speech recognition and speech translation. It includes end-to-end workflows and state-of-the-art models with scalablity and extensibility design. It seamlessly integrates FAIRSEQ's machine translation models and language models to improve S2T model performance. FAIRSEQ S2T documentation and examples are available at https://github.com/pytorch/fairseq/tree/master/examples/speech_to_text.

Acknowledgments

We thank Myle Ott, Michael Auli, Alexei Baevski, Jiatao Gu, Abdelrahman Mohamed and Javad Dousti for helpful discussions.

References

Antonios Anastasopoulos and David Chiang. 2018. Tied multitask learning for neural speech translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,*

[5] From a wav2vec model pre-trained on LibriSpeech: https://github.com/pytorch/fairseq/tree/master/examples/wav2vec

Volume 1 (Long Papers), pages 82–91, New Orleans, Louisiana. Association for Computational Linguistics.

Ebrahim Ansari, Amittai Axelrod, Nguyen Bach, Ondřej Bojar, Roldano Cattoni, Fahim Dalvi, Nadir Durrani, Marcello Federico, Christian Federmann, Jiatao Gu, Fei Huang, Kevin Knight, Xutai Ma, Ajay Nagesh, Matteo Negri, Jan Niehues, Juan Pino, Elizabeth Salesky, Xing Shi, Sebastian Stüker, Marco Turchi, Alexander Waibel, and Changhan Wang. 2020. FINDINGS OF THE IWSLT 2020 EVALUATION CAMPAIGN. In *Proceedings of the 17th International Conference on Spoken Language Translation*, pages 1–34, Online. Association for Computational Linguistics.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, et al. 2019a. Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv preprint arXiv:1907.05019*.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019b. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323.

Sameer Bansal, Herman Kamper, Karen Livescu, Adam Lopez, and Sharon Goldwater. 2019. Pre-training on high-resource speech recognition improves low-resource speech-to-text translation. 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 58–68, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6224–6228. IEEE.

Alexandre Bérard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. 2016. Listen and translate: A proof of concept for end-to-end speech-to-text translation. *arXiv preprint arXiv:1612.01744*.

Dogan Can, Victor R. Martinez, Pavlos Papadopoulos, and Shrikanth S. Narayanan. 2018. Pykaldi: A python wrapper for kaldi. In *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE International Conference on*. IEEE.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE.

Colin Cherry and George Foster. 2019. Thinking slow about latency evaluation for simultaneous machine translation. *arXiv preprint arXiv:1906.00048*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

M. A. Di Gangi, M. Negri, and M. Turchi. 2019. One-to-many multilingual end-to-end speech translation. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 585–592.

Mattia A Di Gangi, Roldano Cattoni, Luisa Bentivogli, Matteo Negri, and Marco Turchi. 2019a. Must-c: a multilingual speech translation corpus. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2012–2017. Association for Computational Linguistics.

Mattia Antonino Di Gangi, Matteo Negri, Roldano Cattoni, Roberto Dessi, and Marco Turchi. 2019b. Enhancing transformer for end-to-end speech-to-text translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 21–31, Dublin, Ireland. European Association for Machine Translation.

Long Duong, Antonios Anastasopoulos, David Chiang, Steven Bird, and Trevor Cohn. 2016. An attentional model for speech translation without transcription. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 949–959.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Charles R Harris, K Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. 2020. Array programming with numpy. *Nature*, 585(7825):357–362.

Hirofumi Inaguma, Shun Kiyono, Kevin Duh, Shigeki Karita, Nelson Yalta, Tomoki Hayashi, and Shinji Watanabe. 2020. ESPnet-ST: All-in-one speech translation toolkit. In *Proceedings of the 58th Annual Meeting of the Association for Computational*

Linguistics: System Demonstrations, pages 302–311, Online. Association for Computational Linguistics.

Jacob Kahn, Ann Lee, and Awni Hannun. 2020. Self-training for end-to-end speech recognition. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7084–7088.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur. 2017. A study on data augmentation of reverberant speech for robust speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5220–5224. IEEE.

Oleksii Kuchaiev, Boris Ginsburg, Igor Gitman, Vitaly Lavrukhin, Carl Case, and Paulius Micikevicius. 2018. Openseq2seq: extensible toolkit for distributed and mixed precision training of sequence-to-sequence models. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 41–46.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Bo Li, Yu Zhang, Tara Sainath, Yonghui Wu, and William Chan. 2019. Bytes are all you need: End-to-end multilingual speech recognition and synthesis with bytes. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5621–5625. IEEE.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yuchen Liu, Jiajun Zhang, Hao Xiong, Long Zhou, Zhongjun He, Hua Wu, Haifeng Wang, and Chengqing Zong. 2020. Synchronous speech recognition and speech-to-text translation with interactive decoding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:8417–8424.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

Xutai Ma, Mohammad Javad Dousti, Changhan Wang, Jiatao Gu, and Juan Pino. 2020a. Simuleval: An evaluation toolkit for simultaneous translation. *arXiv preprint arXiv:2007.16193*.

Xutai Ma, Juan Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2020b. Monotonic multihead attention. *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, Conference Track Proceedings*.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2018. Mixed precision training. In *International Conference on Learning Representations*.

Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. 2019. Transformers with convolutional context for asr. *arXiv preprint arXiv:1904.11660*.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook FAIR's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 314–319, Florence, Italy. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *Interspeech 2019*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.

Juan Pino, Qiantong Xu, Xutai Ma, Mohammad Javad Dousti, and Yun Tang. 2020. Self-training for end-to-end speech translation. *arXiv preprint arXiv:2006.02490*.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. 2011. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, CONF. IEEE Signal Processing Society.

Vineel Pratap, Awni Hannun, Qiantong Xu, Jeff Cai, Jacob Kahn, Gabriel Synnaeve, Vitaliy Liptchinsky, and Ronan Collobert. 2018. wav2letter++: The fastest open-source speech recognition system. *CoRR*, abs/1812.07625.

Colin Raffel, Minh-Thang Luong, Peter J Liu, Ron J Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *International Conference on Machine Learning*, pages 2837–2846.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. 2019. wav2vec: Unsupervised Pre-Training for Speech Recognition. In *Proc. Interspeech 2019*, pages 3465–3469.

Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia X Chen, Ye Jia, Anjuli Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, et al. 2019. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*.

Gabriel Synnaeve, Qiantong Xu, Jacob Kahn, Edouard Grave, Tatiana Likhomanenko, Vineel Pratap, Anuroop Sriram, Vitaliy Liptchinsky, and Ronan Collobert. 2019. End-to-end asr: from supervised to semi-supervised learning with modern architectures. *arXiv preprint arXiv:1911.08460*.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2Tensor for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 193–199, Boston, MA. Association for Machine Translation in the Americas.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Changhan Wang, Kyunghyun Cho, and Jiatao Gu. 2020a. Neural machine translation with byte-level subwords. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:9154–9160.

Changhan Wang, Anirudh Jain, Danlu Chen, and Jiatao Gu. 2019. VizSeq: a visual analysis toolkit for text generation tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 253–258, Hong Kong, China. Association for Computational Linguistics.

Changhan Wang, Juan Pino, and Jiatao Gu. 2020b. Improving cross-lingual transfer learning for end-to-end speech recognition with speech translation. *arXiv preprint arXiv:2006.05474*.

Changhan Wang, Anne Wu, and Juan Pino. 2020c. Covost 2 and massively multilingual speech-to-text translation. *arXiv e-prints*, pages arXiv–2007.

Anne Wu, Changhan Wang, Juan Pino, and Jiatao Gu. 2020. Self-supervised representations improve end-to-end speech translation. *arXiv preprint arXiv:2006.12124*.

Thomas Zenkel, Matthias Sperber, Jan Niehues, Markus Müller, Ngoc-Quan Pham, Sebastian Stüker, and Alex Waibel. 2018. Open source toolkit for speech to text translation. *The Prague Bulletin of Mathematical Linguistics*, 111(1):125–135.

Albert Zeyer, Tamer Alkhouli, and Hermann Ney. 2018. RETURNN as a generic flexible neural toolkit with application to translation and speech recognition. In *Proceedings of ACL 2018, System Demonstrations*, pages 128–133, Melbourne, Australia. Association for Computational Linguistics.

NLPStatTest: A Toolkit for Comparing NLP System Performance

Haotian Zhu **Denise Mak** **Jesse Gioannini** **Fei Xia**
University of Washington, Seattle, USA
{haz060, dpm3, jessegio, fxia}@uw.edu

Abstract

Statistical significance testing centered on p-values is commonly used to compare NLP system performance, but p-values alone are insufficient because statistical significance differs from practical significance. The latter can be measured by estimating effect size. In this paper, we propose a three-stage procedure for comparing NLP system performance and provide a toolkit, NLPStatTest, that automates the process. Users can upload NLP system evaluation scores and the toolkit will analyze these scores, run appropriate significance tests, estimate effect size, and conduct power analysis to estimate Type II error. The toolkit provides a convenient and systematic way to compare NLP system performance that goes beyond statistical significance testing.

1 Introduction

In the field of natural language processing (NLP), the common practice is to use statistical significance testing [1] to demonstrate that the improvement exhibited by a proposed system over the baseline reflects meaningful differences, not happenstance (Dror et al., 2018, 2020). The American Statistical Association emphasizes that ''a p-value, or statistical significance, does not measure the size of an effect or the importance of a result" (Wasserstein and Lazar, 2016). In other words, statistical significance is different from practical significance. The latter is rarely discussed in the NLP field.

To address this issue, we propose a three-stage procedure for comparing NLP system performance, shown in Figure 1. The first stage is building an NLP system and using *prospective power analysis* to compute an appropriate sample size for test corpus. The second stage is hypothesis testing. We stress the need for data analysis to verify assumptions made by significance tests and the importance of estimating the effect size and conducting power analysis. The last stage is to report various results produced by the second stage.

To automate the process, we provide a toolkit, NLPStatTest. We introduce the three-stage comparison procedure (§2), and then describe the the main components (§3) and implementation details (§4) of NLPStatTest. We also present experimental results for running the system on both real-world and simulated data (§5). Lastly, we compare NLPStatTest with existing statistical testing toolkits (§6).

2 Comparing NLP System Performance

In this section we briefly describe the three-stage comparison procedure and define terms that are relevant to NLPStatTest. More detail about Stage 2 can be found in §3-§4.

2.1 Building an NLP System

The first stage is to build an NLP system, run it on test data, and compare the system output with a gold standard. The output of this stage is a list of numerical values such as accuracy or F-scores.

Definition 1 (Evaluation unit). Let (x_j, y_j) be a test instance. An evaluation unit (EU) $e = \{(x_j, y_j), j = 1, \cdots, m\}$ is a set of test instances on which an evaluation metric can be meaningfully defined. A test set is a set of EUs.

Definition 2 (Evaluation metric). Given an NLP system A, the evaluation metric M is a function that maps an EU e to a numerical value:

$$M_A(e) = M\Big(\{(\hat{y}_j, y_j),\ j = 1, \cdots, m\} \Big) \quad (1)$$

where $\hat{y}_j = A(x_j)$ is the system output of A given x_j, and m is the size of e (i.e., the number of test instances in e).

[1] Here we adopt the frequentist approach to hypothesis testing. The debate over frequentist and Bayesian is beyond the scope of this paper.

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 40–46

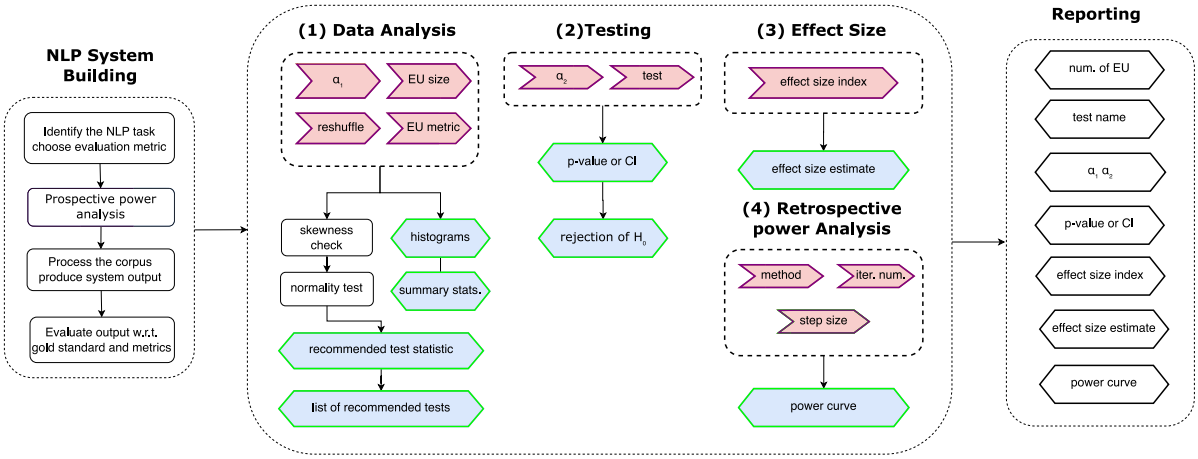

Figure 1: The three-stage procedure for comparing NLP system performance. The pink flag boxes are the parameters that users can either set or use the default values provided by `NLPStatTest`. The blue hexagons are system output of `NLPStatTest`. α_1 and α_2 are the significance levels for normality test and statistical significance test respectively. EU stands for evaluation unit.

An EU may contain one or more test instances. For example, a BLEU score can be computed on one or more sentences. The EU size affects sample size, p-value, sample standard deviation, effect size and so on. It is therefore one of the parameters that users can set when using `NLPStatTest`.

2.2 The Comparison Stage

The second stage is the comparison stage which has four steps (see the largest box in Figure 1).

2.2.1 Data Analysis

When we compare two NLP systems A and B, the output of Stage 1 is a set of pairs, $\{(M_A(e_i), M_B(e_i))\}$, where e_i is the i^{th} EU, and $M_A(e)$ (similarly $M_B(e)$) is defined in Equation 1.

Many statistical tests make certain assumptions about the sample (e.g., normality for t test), so it is important to conduct data analysis to verify those assumptions in order to choose significance tests that are appropriate for a particular sample. If the sample does not follow any known distribution, non-parametric tests should be used.

`NLPStatTest`will estimate sample skewness and test for normality. Then `NLPStatTest` will choose a test statistic (mean or median) for users and recommend a list of significance tests.

2.2.2 Statistical Significance Testing

The second step in Stage 2 is statistical significance testing, using two mutually exclusive hypotheses: the null hypothesis H_0 and the alternative H_1. To compare two NLP systems, a (paired) two-sample test is usually used, though one-sample testing of pairwise difference is equivalent. `NLPStatTest` currently only considers paired two-sample testing for numerical data. Observations within a sample are assumed to be independent and identically distributed ($i.i.d.$).

To run a significance test, users first choose the direction of the test: left-sided, right-sided or two-sided. Then, users specify the hypothesized value of test statistic difference δ and the significance level α, which is often set to 0.05 or 0.01 in the NLP field, and choose a test from the list. `NLPStatTest` will calculate the p-value and reject H_0 if and only if $p < \alpha$.

2.2.3 Effect Size Estimation

In most experimental NLP papers employing significance testing, the p-value is the only quantity reported. However, the p-value is often misused and misinterpreted. For instance, statistical significance is easily conflated with practical significance; as a result, NLP researchers often run significance tests to show that the performances of two NLP systems are different (i.e., statistical significance), without measuring the degree or the importance of such a difference (i.e., practical significance).

Cohen (1990) noted *"the null hypothesis, if taken literally, is always false in the real world."* For instance, because evaluation metric values of two NLP systems on a test set are almost never exactly the same, H_0 that two systems perform equally is (almost) always false. When H_0 is false, the p-value will eventually approach zero in large samples (Lin et al., 2013). In other words, no mat-

ter how tiny the system performance difference is, there is always a large enough dataset on which the difference is statistically significant. Therefore, statistical significance is markedly different from practical significance.

One way to measure practical significance is by estimating *effect size*, which is defined as the degree to which the 'phenomenon' is present in the population: the degree to which the null hypothesis is false (Cohen, 1994). While the need to estimate and report effect size has long been recognized in other fields (Tomczak and Tomczak, 2014), the same is not true in the NLP field. We include several methods for estimating effect size in `NLPStatTest` (see §3.3).

2.2.4 Power Analysis

There are two types of errors in hypothesis testing: Type I errors (false positives) and Type II errors (false negatives). The Type I error of a significance test, often denoted by α, is the probability that, when H_0 is true, H_0 is rejected by the test. The Type II error of a significance test, usually denoted by β, is the probability that under H_1, H_1 is rejected by the test. While Type I error can be controlled by predetermining the significance level, Type II error can be controlled or estimated by power analysis.

Definition 3 (Statistical power). The power of a statistical significance test is the probability that under H_1, H_0 is correctly rejected by the test. The power of a test is $1 - \beta$.

Higher power means that statistical inferences are more correct and accurate (Perugini et al., 2018). While power analysis is rarely used in the NLP field, it is considered good or standard practice in some other scientific fields such as psychology and clinical trials in medicine (Perugini et al., 2018). We implement two methods of conducting power analysis in `NLPStatTest`(see §3.4).

2.3 Reporting Test Results

Beyond the p-value, it is important to report other quantities to make the studies reproducible and available for meta-analysis, including the name of significance test used, the predetermined significance level α, effect size estimate/estimator, the sample size, and statistical power.

3 System Design

`NLPStatTest` is a toolkit that automates the comparison procedure. It has four main steps, shown in the large box in Figure 1. To use `NLPStatTest`, users provide a data file with the NLP system performance scores produced in Stage 1. `NLPStatTest` will prompt users to either modify or use the default values for the parameters in the pink flags and then produce the output in the blue hexagons. The users can then report (some of) the output in Stage 3 of the comparison procedure.

3.1 Data Analysis

The first step of the comparison stage is data analysis, and a screenshot of this step is shown in Figure 2. The top part (above the *Run* button in the purple box) shows the input and parameters that the user needs to provide, and the bottom part (below the *Run* button in the green box) shows the output of the data analysis step.

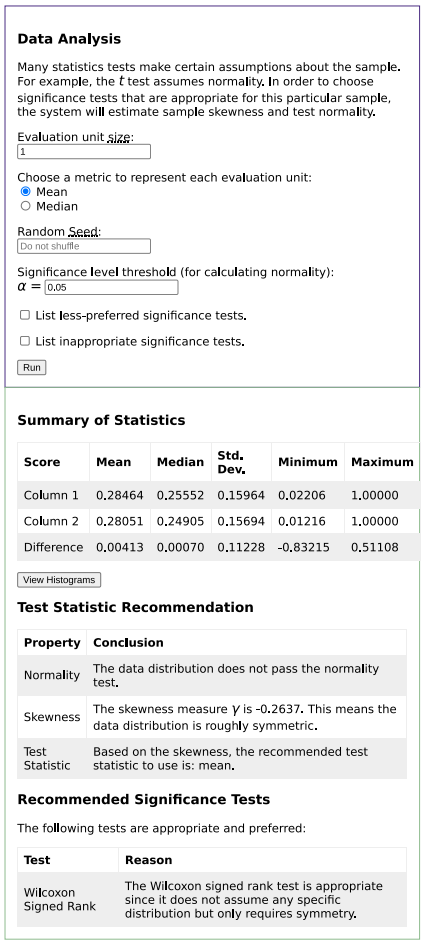

Figure 2: Screenshot of the data analysis step. The part above the *Run* button are parameters that users can set, and the part below is `NLPStatTest` output.

3.1.1 The Input Data File

To compare two NLP systems, A and B, users need to provide a data file where each line is a pair of numerical values. There are two scenarios. In the first scenario, the pair is (u_i, v_i), where $u_i = M_A(e_i)$ is the evaluation metric value (e.g., accuracy or F-score) of an EU e_i given System *A* (see Equation 1), and $v_i = M_B(e_i)$.

In the second scenario, if u_i and v_i can be calculated as the mean or the median of the evaluation metric values of test instances in e_i, users can upload a data file where each line is a pair of (a_k, b_k), where a_k and b_k are the evaluation metric values of a test instance t_k given System *A* and *B*, respectively. Users then chooses the EU size m and specifies whether the EU metric value should be calculated as the mean or the median of the metric values of the instances in the EU. `NLPStatTest` will use m adjacent lines in the file to calculate u_i and v_i. If users prefers to randomly shuffle the lines before calculating u_i and v_i, they can provide a seed for random shuffling.

3.1.2 Histograms and Summary Statistics

From the (u_i, v_i) pairs, `NLPStatTest` generates descriptive summary statistics (e.g., mean, median, standard deviation) and histograms of three datasets, $\{u_i\}$, $\{v_i\}$, and $\{u_i - v_i\}$, as shown in the first table and the three histograms in Figure 2.

3.1.3 Central Tendency Measure

Many statistical tests (t test, bootstrap test based on t ratios, etc) are based on the mean as the test statistic, drawing inferences on average system performance. However, when the data distribution is not symmetric, the mean does not properly measure the central tendency. In that case, the median is a more robust measure. Another issue associated with mean is that if the distribution is heavy-tailed (e.g., the t and Cauchy distributions), the sample mean oscillates dramatically.

In order to examine the symmetry of the underlying distribution, `NLPStatTest` checks the skewness of $\{u_i - v_i\}$ by estimating the sample skewness (γ). Based on the γ value, we use the following rule of thumb (Bulmer, 1979) to determine whether `NLPStatTest` would recommend the use of mean or median as the test statistic for statistical significance testing:

- $|\gamma| \in [0, 0.5)$: roughly symmetric (use mean)
- $|\gamma| \in [0.5, 1)$: slightly skewed (use median)
- $|\gamma| \in [1, \infty)$: highly skewed (use median)

3.1.4 Normality Test

To choose a good significance test for $\{u_i - v_i\}$, we need to determine if the data is normally distributed. If it is, t test is the most appropriate (and powerful) test; if not, then non-parametric tests which do not assume normality might be more appropriate.

If a distribution is skewed according to γ, there is no need to run normality test as the data is not normally distributed. For a non-skewed distribution, `NLPStatTest` will run the Shapiro-Wilk normality test (Shapiro and Wilk, 1965), which is itself a test of statistical significance. The user can choose the significance level (α_1 in Figure 1).

3.1.5 Recommended Significance Tests

Based on the skewness check and normality test result, `NLPStatTest` will automatically choose a test statistic (mean or median) and recommend a list of appropriate significance tests (e.g., t test if $\{u_i - v_i\}$ is normally distributed).

3.2 Testing

In this step, the user sets the significance level (α_2 in Figure 1) and chooses a significance test from the ones recommended in the previous step. If the test has any parameter (e.g., the number of trials for bootstrap testing B), `NLPStatTest` will suggest a default value which can be changed by users. `NLPStatTest` will then run the test, calculate a p-value (and/or provide a confidence interval), and reject H_0 if $p < \alpha_2$.

3.3 Effect Size

Effect size can be estimated by different *effect size indices*, depending on the data types (numerical or categorical) and significance tests. Dror et al. (2020) defined effect size as the unstandardized difference between system performance, while Hauch et al. (2012) and Pimentel et al. (2019) used the standardized difference.

`NLPStatTest` implements the following four indices. Once users select one or more, `NLPStatTest` will calculate effect size accordingly and display the results.

Cohen's *d* estimates the standardized mean difference by

$$d = \frac{\hat{u} - \hat{v}}{\hat{\sigma}} \quad (2)$$

where $\hat{v}$ and $\hat{u}$ are the sample means and $\hat{\sigma}$ denote standard deviation of $u - v$. Cohen's d assumes

normality and is one of the most frequently used effect size indices. If Cohen's d, or any other effect size indices depending on $\hat{\sigma}$, is used to estimate effect size, the EU size will affect the standard deviation and thus effect size estimate.

Hedges' g adjusts the bias brought by Cohen's d in small samples by the following:

$$g = d \cdot \left(1 - \frac{3}{4n - 9}\right) \tag{3}$$

where n is the size of $\{u_i - v_i\}$.

Wilcoxon r is an effect size index for the Wilcoxon signed rank test, calculated as $r = \frac{Z}{\sqrt{n}}$, where

$$Z = \frac{W - n(n+1)/4}{\sqrt{\frac{n(n+1)(2n+1)}{24} - \frac{\sum_{t \in T} t^3 - t}{48}}} \tag{4}$$

Here, W is the test statistic for Wilcoxon signed rank test and T is the set of tied ranks.

Hodges-Lehmann Estimator (Hodges and Lehmann, 1963) is an estimator for the median. Let $w_i = u_i - v_i$. The HL estimator for one-sample testing is given by

$$HL = \text{median}\left(\{(w_i + w_j)/2,\ i \neq j\}\right) \tag{5}$$

3.4 Power Analysis

Power (Definition 3) covaries with sample size, effect size and the significance level α. In particular, power increases with larger sample size, effect size, and α. There are two common types of power analysis, namely *prospective* and *retrospective power analysis*, and `NLPStatTest` implements both types.

3.4.1 Prospective Power Analysis

Prospective power analysis is used when planning a study (usually in clinical trials) in order to decide how many subjects are needed. In the NLP field, when one constructs or chooses a test corpus for evaluation, it will be beneficial to conduct this type of power analysis to determine how big a corpus needs to be in order to ensure that the significance test reaches the desired power level.

In `NLPStatTest`, prospective power analysis is a preliminary and optional step. The user needs to provide the expected mean and standard deviation of the differences between samples, the desired power level, and the required significance level. `NLPStatTest` will calculate the minimally required sample size for t test via a closed form, assuming the normal distribution of the data.

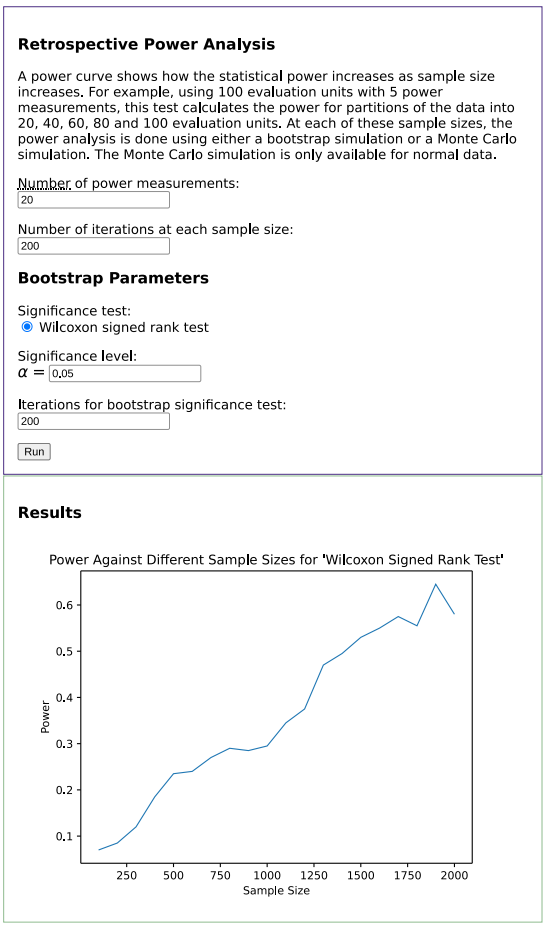

Figure 3: Screenshot for retrospective power analysis.

3.4.2 Retrospective Power Analysis

Retrospective or *post-hoc power analysis* is usually done after a significance test to determine the relation between sample size and power. There are two scenarios associated with retrospective power analysis: When the values in $\{u_i - v_i\}$ are from a known distribution, one can use *Monte Carlo simulation* to directly simulate from this known distribution. To do this, one has to have an informed guess of the desired effect size (i.e., mean difference) via meta-analysis of previous studies.

When the distribution of the sample is unknown *a priori*, one can resample with replacement from the empirical distribution of the sample (a.k.a. the *bootstrap* method (Efron and Tibshirani, 1993)) to estimate the power.

`NLPStatTest` implements both methods. Users can employ one or both; `NLPStatTest`

will produce a figure that shows the relation between sample size and power, as in Figure 3.

4 Implementation Details

The `NLPStatTest` graphical user interface can be run locally or on the Web. There is also a command line version. The graphical tool, the command line tool, the source code, a user manual, a tutorial video are available at `nlpstats.ling.washington.edu`. We recommend using an updated Chromium-based browser.

The client-side web interface is written in HTML, CSS, and JavaScript (with JQuery). The server-side code is written in Python, using the Flask web framework. YAML is used for configuration files. KaTeX is used to render mathematical symbols. The Python code uses the SciPy and NumPy libraries to implement statistical tests and Matplotlib to generate the histograms and graphs.

5 Experiment

To test the output validity and speed of `NLPStatTest`, we run experiments using both real and simulated data.

5.1 Real Data from WMT-2017

The WMT-2017 shared task (Bojar et al., 2017) reported system performance results based on human evaluation scores; unpaired testing (Wilcoxon rank-sum) was used because not many sentences had human evaluation scores for both MT systems that were being compared.

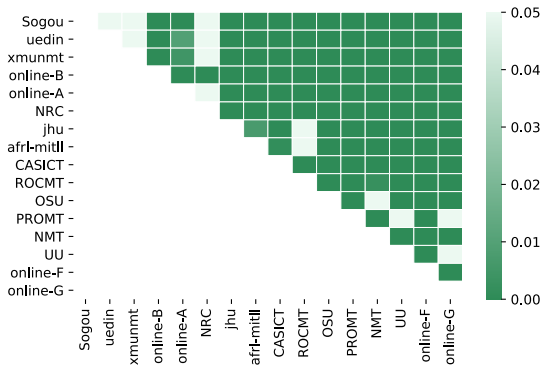

Figure 4: Heatmap of pairwise comparison for the 16 WMT-2017 Chinese-to-English MT systems. BLEU scores and Wilcoxon signed-rank test are used. p-values are adjusted via Bonferroni correction. Dark green cells indicate statistical significance ($p < 0.05$); light green cells indicate non-significance ($p \geq 0.05$).

Because `NLPStatTest` currently implements paired testing only, we use the Wilcoxon signed-rank test (instead of Wilcoxon rank-sum test) and the BLEU scores (instead of human evaluation scores) when comparing MT systems. According to Bojar et al. (2017), a set of 15 or more sentence-level evaluation scores constitutes a reliable measure of translation quality; thus, we set the EU size to be 15. We also reshuffled the scores before grouping test instances into evaluation units.

Figure 4 shows the results of pairwise comparisons among all 16 Chinese-to-English MT systems (120 system pairs in total). The heatmap is similar to the comparison results in Bojar et al. (2017) (see Figure 5 in that paper). The minor differences of the two heatmaps are due to different evaluation metrics (BLEU vs. human scores), the significant tests (Wilcoxon signed-rank vs. Wilcoxon rank-sum), and the numbers of EUs (more test sentences have BLEU scores than human evaluation scores).

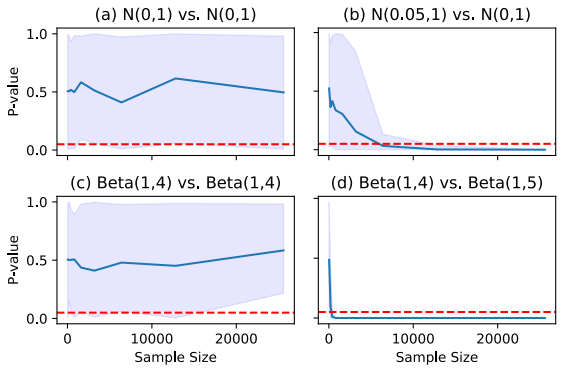

Figure 5: Plots of p-value against sample size. Figure (a) and (b) use two samples with normal distribution, while (c) and (d) use Beta distribution. H_0 should be true for (a) and (c) and false for (b) and (d). We run t test for (a) and (b), and Wilcoxon signed-rank test for (c) and (d). The red dotted line stands for the threshold $\alpha = 0.05$. The light purple shade depicts the range of p-values. The solid blue line denotes the mean of p-values for each sample size.

5.2 Simulated Data

We also run simulation experiments on `NLPStatTest` to validate the testing results. Here, we conduct two-sided, paired testing, varying sample size from 30 to 25,000, each with 20 iterations of tests to obtain a range of p-values. As shown in Figure 5, when H_0 is true (see Fig 5(a) and 5(c)), p-values range freely in $(0, 1)$. When H_0 is false (see 5(b) and 5(d)), p-values approach zero as sample size increases,

as expected. The fast convergence to zero in 5(d) may be due to the small variance of the differences between the two Beta samples (≈ 0.046), even though the difference between sample medians is small (≈ 0.02). In contrast, 5(b) converges to zero much more slowly due to the large variance.

6 Related Work

Dror et al. (2018) made an accompanying package available [2] for hypothesis testing. This package includes functionalities such as testing for normality, t testing, permutation/bootstrap testing, and using McNemar's test for categorical data. `NLPStatTest` implements all the aforementioned tests except McNemar's test. In addition, `NLPStatTest` offers data analysis, effect size estimation, power analysis and graphical interface.

`NLPStatTest` is based on the frequentist approach to hypothesis testing. Sadeqi Azer et al. (2020) developed a Bayesian system[3] which uses the Bayes factor to determine the posterior probability of H_0 being true or false.

7 Conclusion

While statistical significance testing has been commonly used to compare NLP system performance, a small p-value alone is not sufficient because statistical significance is different from practical significance. To measure practical significance, we recommend estimating and reporting of effect size. It is also necessary to conduct power analysis to ensure that the test corpus is large enough to achieve a desirable power level. We propose a three-stage procedure for comparing NLP system performance, and provide a toolkit, `NLPStatTest`, to automate the testing stage of the procedure. For future work, we will extend this work to hypothesis testing with multiple datasets or multiple metrics.

References

O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, S. Huang, M. Huck, P. Koehn, Q. Liu, V. Logacheva, C. Monz, M. Negri, M. Post, R. Rubino, L. Specia, and M. Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

M. G. Bulmer. 1979. *Principles of Statistics*, page 57. Dover, New York.

J. Cohen. 1990. Things I have learned (so far). *American Psychologist*, 45(12):1304 – 1312.

J. Cohen. 1994. The earth is round ($p < .05$). *American Psychologist*, pages 997–1003.

R. Dror, G. Baumer, S. Shlomov, and R. Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of ACL-2018 (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia.

R. Dror, L. Peled-Cohen, S. Shlomov, and R. Reichart. 2020. *Statistical Significance Testing for Natural Language Processing*. Morgan & Claypool.

B. Efron and R. J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall, New York, NY.

V. Hauch, I. Blandón-Gitlin, J. Masip, and S. L. Sporer. 2012. Linguistic cues to deception assessed by computer programs: A meta-analysis. In *Proceedings of the Workshop on Computational Approaches to Deception Detection*, pages 1–4, Avignon, France.

J. L. Hodges and E. L. Lehmann. 1963. Estimates of location based on rank tests. *Annals of Mathematical Statistics*, 34(2):598–611.

M. Lin, H. Lucas, and G. Shmueli. 2013. Too big to fail: Large samples and the p-value problem. *Information Systems Research*, 24:906–917.

M. Perugini, M. Gallucci, and G. Costantini. 2018. A practical primer to power analysis for simple experimental designs. *International Review of Social Psychology*, 31.

T. Pimentel, A. D. McCarthy, D. Blasi, B. Roark, and R. Cotterell. 2019. Meaning to form: Measuring systematicity as information. In *Proceedings of ACL-2019*, pages 1751–1764, Florence, Italy.

E. Sadeqi Azer, D. Khashabi, A. Sabharwal, and D. Roth. 2020. Not all claims are created equal: Choosing the right statistical approach to assess hypotheses. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

S. S. Shapiro and M. B. Wilk. 1965. An analysis of variance test for normality (complete samples)†. *Biometrika*, 52(3-4):591–611.

M. Tomczak and E. Tomczak. 2014. The need to report effect size estimates revisited. An overview of some recommended measures of effect size. *Trends in Sport Sciences*, 21:19–25.

R. L. Wasserstein and N. A. Lazar. 2016. The ASA statement on p-values: Context, process, and purpose. *The American Statistician*, 70(2):129–133.

[2] https://github.com/rtmdrr/testSignificanceNLP

[3] https://github.com/allenai/HyBayes

Author Index