MULTIMODALQA - COMPLEX QUESTION ANSWERING OVER TEXT, TABLES AND IMAGES - SUPPLEMENTARY MATERIAL

Anonymous authors
Paper under double-blind review

2 DATASET GENERATION

In this section we provide more details on how we extracted tables from Wikipedia, parsed them, enriched them with images and text questions, generated single modality questions and paraphrased our machine-generated questions.

2.1 WIKIPEDIA TABLES AS ANCHORS

Extraction As specified in the main paper we use the 01-01-2020 English Wikipedia dump of Wikipedia that contains roughly $3M$ tables. From the said dump, we extracted all tables and selected those that meet the following criteria: (a) The tables contain $10 - 25$ rows, i.e. they are not too short or too long; (b) At least three images are associated with the table. This results in a total of $700K$ remaining tables.

To extract tables from Wikipeida we created a customized version of the publicly available WikiTextParser package [1]. The customization process included adding the ability to extract table titles and modifying issues regarding rows and columns spans.

Classifying Table Columns During question generation, some methods in our logical language require certain restrictions on table columns. For example, $\text{COMPARE}(\cdot, \cdot)$ requires a numeric or date column to exist in the table. Hence, when we extract tables we also classify columns in the tables based on the data they incorporate. We define 3 *semantic types* for columns: numeric, date and index. The classification is built such that: (1) a column for which all the cells can be parsed into date objects is classified as a *date column*; (2) a column for which all the cells can be parsed into numeric values is classified as a *numeric column*; (3) a *numeric column* with consecutive values is classified as an *index column*.

2.2 CONNECTING IMAGES AND TEXT TO TABLES

2.2.1 IMAGES

We elaborate on two cases: (a) in-table images and (b) images from pages of linked *WikiEntities*.

In-table images Some tables feature images inside the table cells. These columns usually appear in tables that include lists of objects, such that another column in the table includes a description of the cells depicted in the images. For example, a Wikipedia table can list buildings of a certain city, such that one column includes the names, or *WikiEntities*, of the buildings, and a different column would contain the buildings' images.

However, it is not given in the table which column provides the description for the images, hence we deploy a linear classifier that aims to assess that. The features of the classifier include the column's distance from the leftmost column, the percentage of unique cells, the percentage of cells with exactly one *WikiEntity*, the percentage of cells with short text (at most 2 chars), and the column's header.

[1] https://github.com/5j9/wikitextparser

THE MEDIAWIKI API We use the MediaWiki API package [2] to retrieve the URL of the image. We assign a unique identifier to each image based on the table, row, and the column associated with it. We store the images in Amazon S3 using the image's identifier.

Images from *WikiEntities* Although some Wikipedia tables include images, this is not the common case. More frequently, a table would contain a column of *WikiEntities* that could potentially be linked to images. Hence, we need to retrieve and associate between *WikiEntities* and their images.

To associate the *WikiEntities* with their representative image, we first go over the entire Wikipedia dump and create a mapping between the *WikiEntities* and their associated image Wikipedia files. Next, we try and retrieve the images' URLs from the Wikipedia file using the MediaWiki API, similar to the in-table images. Afterwards, each image is stored in S3 with the matching entities name, and identified using the *WikiEntity* title. Because each image is associated with a specific *WikiEntity*, the image is cached and can be used across several contexts.

The process of matching *WikiEntities* to their representative image cannot be done for all entities. For several reasons: (i) files in Wikipedia can be removed over time, which means that we will not be able to successfully retrieve an image for every mapped entity; (ii) some *WikiEntities* do not have an image associated with them in the Wikipedia dump; (iii) Wikipedia editors do not always link table cells to *WikiEntities*; (iv) some images are not representative of the *WikiEntities* they depict. For example, certain film entities are depicted using the same film logo. This image is not descriptive of a specific film, and cannot be used during question generation. In order to avoid associating *WikiEntities* with non-descriptive images, we created a list of frequently used non-representative images, and did not link images that were included in this list.

Overall, we obtain 48,885 images, with 671 in-table images and 48,214 *WikiEntity* images.

2.3 GENERATING SINGLE-MODALITY QUESTIONS

2.3.1 GENERATING IMAGE QUESTIONS

Single Images When generating single-image questions, we show Amazon Mechanical Turk (AMT) crowdworkers an image, alongside its *WikiEntity*, and ask them to phrase a question about the image with the entity being the focus of the question.

The User Interface (UI) is depicted in figure 1. As can be seen the AMT worker is shown an image and a *WikiEntity* ([Statue of Liberty]), and is asked to generate an appropriate question, and its matching answer, e.g. *"What is the [Statue of Liberty] holding in her right hand?"*. Additionally, the UI has a comments input, to communicate any thoughts, concerns or questions. We have pursued communication with AMT workers based on the comments they left us.

To confirm that the question is suitable in an open-domain setting, we dissuade workers from asking questions about a temporary, i.e. "non-stable", features of the image. For instance, the question *"What is the [Statue of Liberty] holding in her right hand?"* is valid in an open-domain setting, since it is unlikely that in other pictures of the the *WikiEntity* the answer to the question would differ. However, a question such as *"What is the color of the sky behind the [Statue of Liberty]?"* could not be used in an open-domain setting, since a picture of the same *WikiEntity* in different times of the day would lead to a different answer.

We note that we apply a-priori filtering on images that will be good candidates for multi-modal questions. We do so by selecting *WikiEntities* that were associated to questions from other modalities.

Post generating the questions, we ask a separate group of AMT workers to verify that the question meets the task's criteria.

[2] https://www.mediawiki.org/wiki/API:Main_page

Figure 1: When generating single-image questions, we show AMT workers an image alongside its *WikiEntity*, and ask them to phrase a question about the image with the entity being the focus of the question.

Image Lists For questions with a list of images, we use images that appear in the same column of a table. To generate these questions, AMT workers were given the images and asked to phrase a binary question about a distinctive feature of the entities that a subset of the images share.

The UI is depicted in figure 2. As can be seen the AMT worker receives a list of images from a single column, and is asked to generate a binary question that applies to all of them and provide its answer. To confirm that the question is suitable for an open-domain setting, we ask the workers to confirm that their question asks about a non-temporary, i.e. stable, feature of the images, such as geographic features or color of a person's eyes.

For the image lists questions to be challenging, we heuristically filter columns that can be used as anchors for the image lists questions. Every such column needs to include at least 4 *WikiEntities*, no more than 3 duplicate images, and no more than 2 *WikiEntities* without images.

Post generating the questions, we ask a separate group of AMT workers to verify that the question meets the generation criteria.

This process results in 2,268 single image questions and 2,223 list image questions that are later used to create multimodal questions.

2.3.2 Generating Text Questions

We leveraged various text Question Answering (QA) Reading Comprehension (RC) datasets in order to obtain single modality free text questions, that are used to compose multimodal questions (*NQ* - Kwiatkowski et al. (2019), *BoolQ* - Clark et al. (2019), *HotpotQA* - Yang et al. (2018)). We describe how we linked the questions in the RC datasets to the anchor tables of our contexts.

Linking text questions to tables In order to link between questions in the RC datasets and the tables, for each question in the RC datasets we applied the following process: (1) we extracted all the *WikiEntities* from the HTML of the question's gold article; (2) we searched for matches between the tokens of each extracted *WikiEntity* and the question using spaCy (Honnibal & Montani (2017)); (3) if a match was found, we marked the *WikiEntity* as one that appears in the question; (4) if a *WikiEntity* is shared between the question and table we connected them. We note that since *BoolQ* and *HotpotQA*

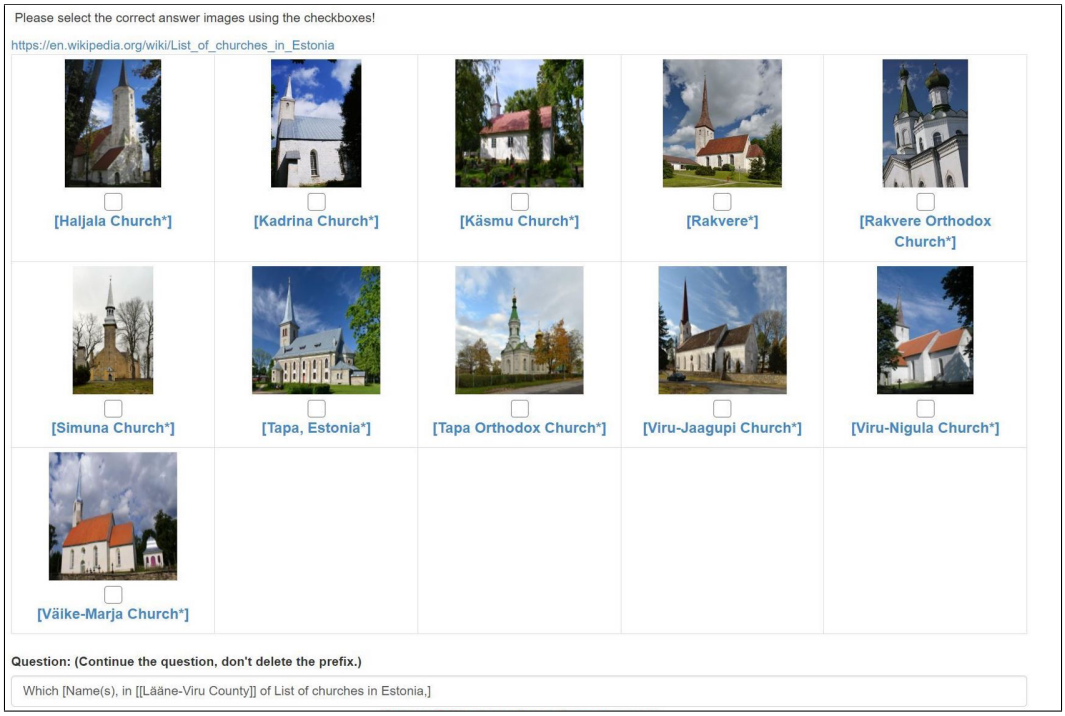

Figure 2: The UI for our image list questions. AMT workers were given images originating from the same column, and were asked to phrase a binary question about a distinctive feature of the entities that a subset of the images share. E.g., given a list of buildings,the worker could ask *"Which of the buildings have a light blue roof?"*

do not provide the HTML of the question's context as part of the dataset, we retrieved the HTML file of each context using the MediaWiki API.

2.5 PARAPHRASING USING AMT

We asked AMT workers to paraphrase automatically-generated pseudo language (PL) questions into natural language. To do so we created a special AMT user interface, that provides various method of feedback as to the quality of paraphrasing.

The UI for the paraphrasing task is depicted in figure 3. As can be seen the AMT workers are presented with a machine generated question, its answer, the 1st hop answer (denoted *"Bridge answer"* in the UI, presented if it exists for the question), and relevant parts of the context. The worker is asked to paraphrase the question such that it maintains its original meaning, but is phrased naturally.

We also deployed a feedback mechanism, where workers receive a bonus if a baseline model correctly answered the question after their first paraphrasing attempt, but incorrectly after they refined the paraphrase. The workers gain access to the model feedback by pressing *"See AI Answer"* button in the UI. To generate diversity, workers got a bonus if the normalized edit distance of a paraphrase compared to the PL question was higher than 0.7, this feedback was presented to them automatically in each rephrase.

Post generating the the questions, we asked a separate group of AMT workers to verify that the question meets the generation criteria. The UI for the validation task is shown in figure 4. The workers receive the original machine generated PL question, the human rephrase, and relevant parts of the context. They are tasked with determining whether the rephrase indeed keeps the same meaning, and as a another measure of quality they rate how likely it is that the rephrase question would have been asked by a human.

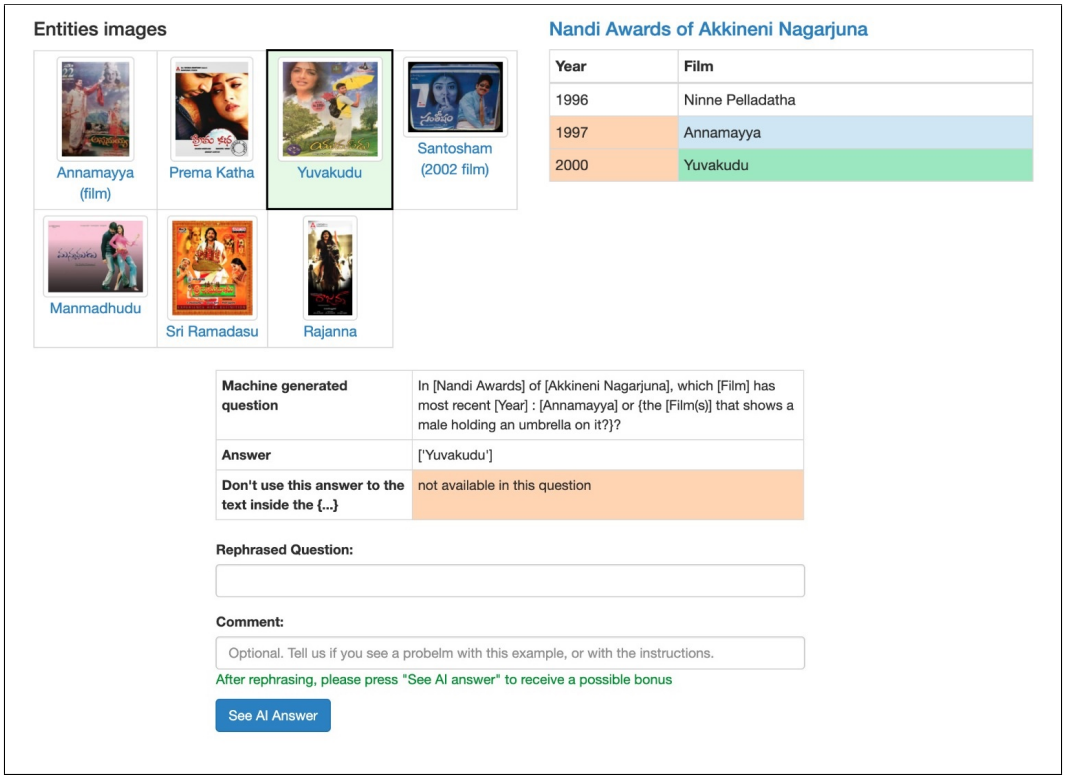

Figure 3: An example of a PL question presented in our UI for the AMT workers during paraphrasing. For each question, we show the PL question alongside relevant context from all the modalities, and ask from the AMT workers to rephrase the PL question in NL.

3 Dataset Analysis

Paraphrasing Richness Crowd workers received a bonus when substantially modifying the automatically-generated questions, to encourage diversity in the natural language questions. Figure 5 shows the normalized edit distance distribution between the NL questions and the PL questions.

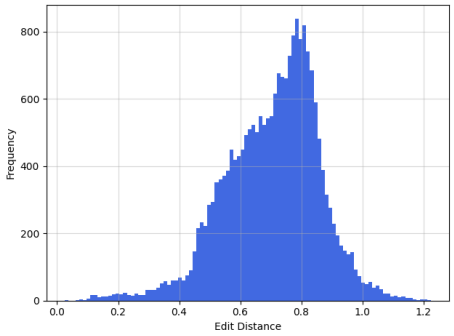

Figure 5: MG and NL edit distance similarity.

4 Models

We provide additional information on some of the modules used and how we applied them.

4.1 Single-Modality QA Modules

Image QA Module Our starting point is VILBERT-MT (Lu et al., 2020; 2019), a model trained on a wide variety of vision-and-language tasks that includes visual question answering, caption-based image retrieval, grounding referring expressions and multimodal verification. The model sees as inputs image region features extracted by a vision network (Faster R-CNN (Ren et al., 2015) with a ResNet-101 backbone (He et al., 2016)) pre-trained on the Visual Genome dataset (Krishna et al., 2017), as in Anderson et al. (2018). From each image, we extracted a 100 2048-dimensional region features. Given the high memory and computational requirements of processing multiple images, we processed each image separately, and combined the answers as described in the main paper.

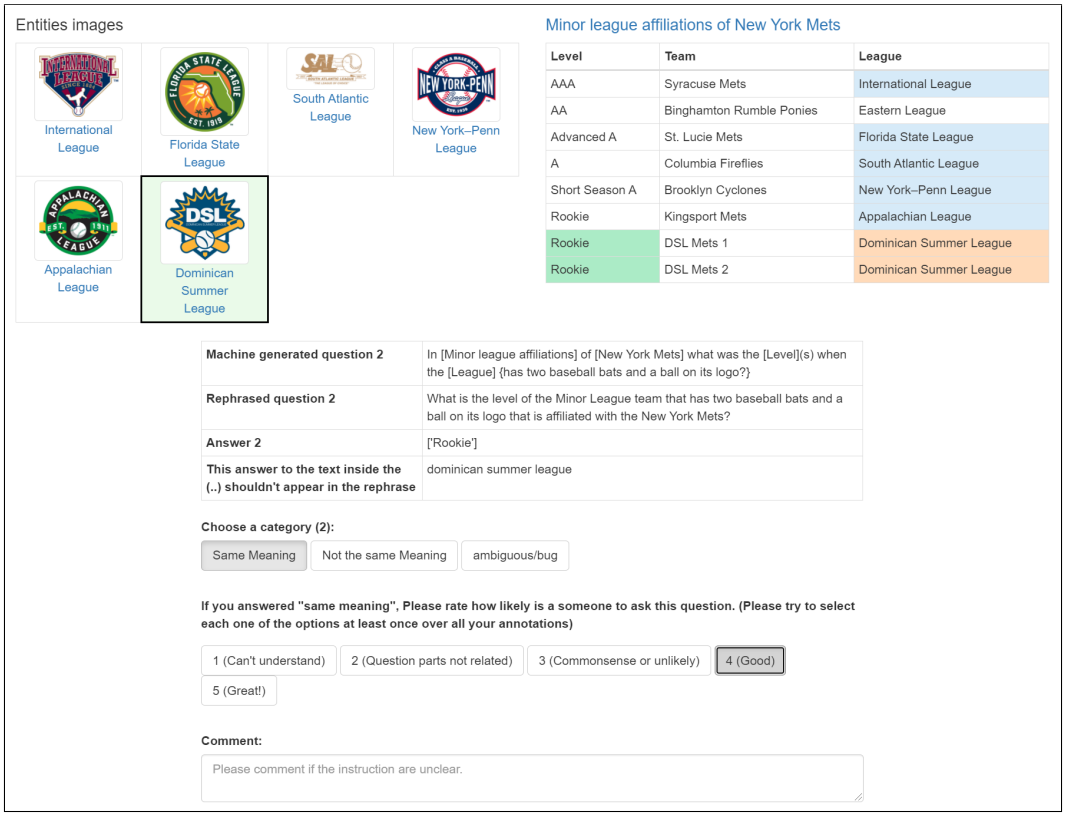

Figure 4: An example for the rephrasing validation task in our UI for the AMT validation workers. For each rephrased question, we show an automatically-generated PL question and its rephrasing along with the relevant context from all the modalities. We ask the AMT validation workers to determine whether the rephrasing maintains the meaning of the question, and how likely it is that the rephrase question would have been asked by a human.

5 EXPERIMENTS

For our baselines we first train a question-type classifier, based on RoBERTa-large, that takes a question Q as input, and predicts one of the 16 possible question types. Our question type classifier obtains an overall accuracy of 91.4% on the test set. Figure 6 shows the classifier's program prediction confusion matrix, such that predictions on the diagonal are correct, and outside the diagonal are incorrect. We observe several incorrect predictions between the programs containing the *Compose()* operator and TextQ, this is assumed to be due to similar questions in the *HotpotQA* that are used as part of TextQ.

We evaluate the human performance using a special UI that enables the user to move between modalities in the context. Figure 7 Illustrates the components in a full context: 10 Text Paragraphs, a list of images and a table. The user may chose on which modality to inspect before providing the final answer. The answer is provided as free text.

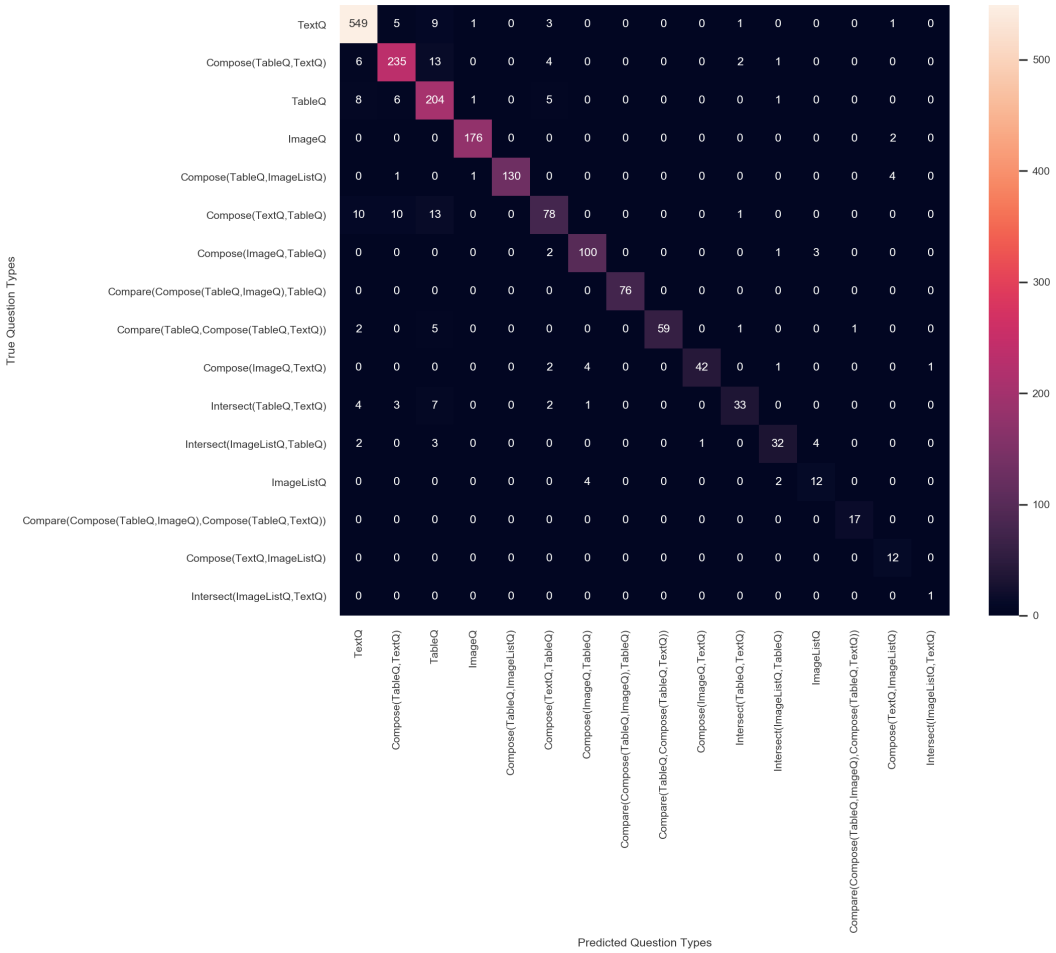

Figure 6: Confusion matrix of the question type classifier on the dev set.

Figure 7: An example for human evaluation performance task. For each question, we show the relevant context from all the modalities, and ask the user to answer the question.

REFERENCES

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6077–6086, 2018.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pp. 13–23, 2019.

Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 12-in-1: Multi-task vision and language representation learning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.