Experiment information extraction (per paper)

General Information			
Name of experiment	PtrGNCMsg	Authors	Qin Liu, Zihe Liu, Hongming Zhu, Hongfei Fan, Bowen Du, Yu Qian
Year of publication	2019	Available online	Available at link - https://zenodo.org/records/2542706#.XECK8C277BJ
Publication paper name	Generating Commit Messages from Diffs using Pointer-Generator Network	Notes on availability	N/A

Experiment Information	
Comparison	
Treatments	Neural Machine Translation - NMT Version 1.1 Neural Machine Translation - NMT Version 1.2 Neural Machine Translation - NMT Version 1.3 Neural Machine Translation - NMT Version 1.4
Ablation study	Not performed
Human evaluation	Not performed
Metrics	BLEU-4, ROUGE-1, ROUGE-2, ROUGE-L
Other variables	No statistical variables used to perform statistical analysis other than the above metrics
Training runs before reporting	Number of runs not specified. Mention that trainings are performed until hyperparameter value is decided, but not for final hyperparameters
Comparison procedure	Comparison tries to answer about accuracy with 0.9 cover rate (final version) and amongst different cover rates. Comparison against NMT is made training both in datasets top 1000 and top 2000 both lowercased and not. Then, a case study on specific outputs is presented to ilustrate why the quality of the proposed approach is better, taking examples of all datasets used
Experimental results	
Main results	PtrGNCMsg performs better than NMT on cover rates below 1 consistently, used 0.9 as cover rate so it performs better than NMT Out of Vocabulary words are better captured by PtrGNCMsg than NMT.
Other results	Cover rate set to 0.9 for best performance compared to other values examined
Experimental Setting	
Hardware	Computer equiped with GTX 1080Ti 12G GPU
Software	TensorFlow 1.9.0 absl-py 0.3.0 / astor 0.7.1 / astroid 2.0.1 / gast 0.2.0 / grpcio 1.13.0 / isort 4.3.4 / javalang 0.11.0 / lazy-object-proxy 1.3.1 / Markdown 2.6.11 / mccabe 0.6.1 / nltk 3.3 / numpy 1.15 / protobuf 3.6 / pylint 2.0.1 / pymongo 3.7.1 / siz 1.11 / tensorboard 1.9 / termcolor 1.1 / typed-ast 1.1 / werkzeug 0.14.1 / wrapt 1.10.11
Test Dataset	Yes, same test set used on both dataset trained. These are top 1000 (first half of dataset) and top 2000 (with the new data addition) both in lowercase and without lowercasing. Test set to compare same origin as in training

General Information			
Name of experiment	ATOM	Authors	Shangqing Liu, Cuiyun Gao, Sen Chen, Lun Yiu Nie, Yang Liu
Year of publication	2022	Available online	Not available - link provided https://github.com/shangqing-liu/ATOM
Publication paper name	ATOM: Commit Message Generation Based on Abstract Syntax Tree and Hybrid Ranking	Notes on availability	Link not working 404 error on GitHub

Experiment Information	
Comparison	
Treatments	Neural Machine Translation - NMT Luong Att - Version 2.1 Neural Machine Translation - NMT Bahdanau Att - Version 2.2 NMT Trained in dataset divided by project - Version 2.3 NMT Trained in dataset divided by timestamp - version 2.4 NNGen - Version 1.1 NNGen used in dataset divided by project - Version 1.2 NNGen used in dataset divided by timestamp - Version 1.3 PtrGNCMsg (named Ptr-Net in the paper) - Version 1 CoDiSum - Version 1 Commit2Vec - Version 1
Ablation study	Ablation study performed on deactivation of generation and retrieval modules. First, generation is deactivating, relying only on retrieval, and after it the opposite is done, relying in generation only. Ablation study is evaluated as part of the treatments as ATOM versions 0.1 and 0.2.
Human evaluation	Human evaluation performed. 6 people invited from researchers and PhD. 100 random commits evaluated on a survey scoring from 0 to 4. For each, provided with reference message and generated from NNGen (v1.1), NMT Luong Att (v2.1) and ATOM.
Metrics	BLEU 1/2/3/4 / ROUGE-L / METEOR
Other variables	Related to the Human evaluation, Average, Standard Deviation and Pearson Correlation Coefficient are used to statistically analyze results. Also percentages used when introducing % of messages output by retrieval or generation. Lastly, count and ratio over all is used when evaluating the performance on different lines on the code diffs
Training runs before reporting	Not mentioned.
Comparison procedure	The process first evaluates based on the metrics the performance of all treatments, including the ablation study performed on ATOM without generation and retrieval modules. Then % of outputs generated by retrieval and generation on last approach is presented. After it, evaluation on different number of added/deleted paths is presented, explaining why the configuration is done at 80. Lastly, on the ranking module, many different methods for ranking are tried and compared using the metrics.
Experimental results	
Main results	ATOM outperforms every other approach all trained on the presented dataset Human evaluation shows ATOM outputs are better than other approaches Encoding using ASTs gives better understanding of semantics than previous approaches used to compare When approaches evaluated in dataset divided by project and timestamp, performance decreases significantly, though ATOM still performs the best.
Other results	ATOM optimal number of paths is 80, similar performance but slightly lower when 100 From evaluated ranking methodologies (XGBoost,SVR,GRU,LTSM with and without Att.) the ConvNet approach performs best ATOM Encoder can handle longer diffs due to AST input Ablation study concludes using both generation and retrieval together is benefitial in terms of performance
Experimental Setting	
Hardware	Server with 36 cores and 4 NVIDIA GPUs models TELSA P40 and M40, 22GB VRAM each
Software	Tensorflow 1.12 Pytorch 1.4 No further specification in paper and not available online to extract more
Test Dataset	Not specifically mentioned. Assumed that main experiment uses same test set. For last part, the evaluate the dataset split by project and by timestamp to evaluate how other dataset divisions might affect and also in case there is leakage of data between sets in original dataset evaluation (possible).

General Information			
Name of experiment	NMT	Authors	Siyuan Jiang, Ameer Armaly, and Collin McMillan
Year of publication	2017	Available online	NOT AVAILABLE - Link provided https://sjiang1.github.io/commitgen
Publication paper name	Automatically Generating Commit Messages from Diffs using NMT	Notes on availability	Link provided but error 404 not found in GitHub when trying to open it

Experiment Information	
Comparison	
Treatments	MOSES - Not aimed for commit generation - Aimed for automatic Translation NMT Ablation study on VDO filter - Versions 0 and 0.1
Ablation study	Ablation study performed on Verb-Direct Object filter effectiveness. Deactivates the VDO filter and creates a new test set to compare both alternatives to check effectiveness of VDO filter. Alternative without the VDO filter receives training on a different training set since it does not remove using VDO filter from dataset preprocessing (x2.5 times more data than NMT) and a different test set. Then both approaches are compared.
Human evaluation	Human evaluation performed on 20 participants with 30 min to evaluate reference and generated messages using a survey which asks to evaluate the message from 0 to 7. 226 pairs evaluated by 3 participants, 522 by 2 participants and remaining 235 by only one participant.
Metrics	BLEU-4 / Modified N Gram Precision for N=1/2/3/4 (p1,p2,p3,p4)
Other variables	For the ablation study and the human evaluation, along with the other aspects analyzed in the paper, the statistical metrics used are counts and median of scores on human evaluation rounded down.
Training runs before reporting	Ran Nematus with settings from the last 4 saved models (4 runs) and then the final model to obtain ensemble results.
Comparison procedure	The comparison and experimental procedure compares firs MOSES and NMT using BLEU and modified n-gram precision, then extract new dataset for ablation study to train and test ablation on VDO filter and compares with first NMT. After it, distribution of metrics is computed for different lengths of code diff as input, also using BLEU and modified n-gram precision. Then human evaluation is conducted. Lastly, a Quality Assurance filter implemented as a SVM trained with SGD to catalogue good and bad commits is applied to results
Experimental results	
Main results	NMT outperforms MOSES by a vast margin on used dataset Many limitations still to generation of commit message according to results of QA filter (43.8% of recall) Ablation study shows that many quality input do not start with Verb or Direct Object Human evaluation shows that model either generates too good or too bad (lots of 0s and 7s in survey) These kind of systems are valid for short commits, not aiming for long and big changes
Other results	Applying QA filter after generation reduces 44% of bad messages in expense of 11% of good messages Human evaluation does not give a clear trend for model generation Removing VDO filter sometimes create commit messages without sense or version numbers that should not be generated Variation with VDO filter has some value but performs much poorer than regular NMT over its dataset
Experimental Setting	
Hardware	Computer with Nvidia GeForce GTX 1070 with 8Gb of VRAM. No further details mentioned.
Software	No mention of software versions used. As they reference Nematus, looking paper and repository, we extract: Python 3.5.2 Tensorflow versions from 1.15 to 2.0
Test Dataset	On comparison against MOSES - Same dataset used for test and same GPU used in test. Test set comes from same origin dataset. On ablation study - new test set created selecting 3,000 pairs randomly from 72,000 pairs remaining of preprocessing without applying the VDO filter to the Jiang and McMillan dataset

General Information			
Name of experiment	CoDiSum	Authors	Shengbin Xu , Yuan Yao , Feng Xu , Tianxiao Gu , Hanghang Tong and Jian Lu
Year of publication	2019	Available online	Available at https://github.com/SoftWiser-group/CoDiSum - Link found not provided
Publication paper name	Commit Message Generation for Source Code Changes	Notes on availability	Lots of aspects not mentioned in the paper are mentioned because they were extracted from link

Experiment Information	
Comparison	
Treatments	All approaches are assumed to be trained in proposed dataset since metrics do not match original papers, but no specific mention about this is made: Neural Machine Translation - NMT - Version 3 NNGen - Version 2 CopyNet - Not aimed for commit generation - Integration of copying mechanism into a decoder
Ablation study	Ablation study on performance gain comparing the following alternatives to main CoDiSum: CoDiSum v0.1 - deletes copying mechanism and jointly model semantics and structure CoDiSum v0.2 - deletes code semantics and replaces identifiers with placeholders to reduce vocab. size CoDiSum v0.3 - Directly concatenates structural and semantical in the input
Human evaluation	No human evaluation performed
Metrics	BLEU-4 (not specified in paper which Ngram but based on results assumed 4) / METEOR / Recall in %
Other variables	No further analysis outside the metrics presented is used to statistically analyze the approach
Training runs before reporting	Not mentioned in the paper or in the repository. By the way of training from repository and the use of seed, assumed 1 run before reporting results
Comparison procedure	For the comparison, all approaches are compared using the metrics (as not many information available, assumed all trained on the proposed dataset). Later, effect of variation on the embedding size and on the hidden state size on the metrics is presented. Last, a case study on various outputs is presented and analyzed by the authors.
Experimental results	
Main results	CoDiSum overperforms the compared approaches on the proposed dataset, but not in every case and not by a large gap One of main contributions compared to other approaches is the inclusion of identifiers from the input into the output
Other results	CoDiSum mantains stable when parameter word embedding size is in range 50-300 Metics BLEU and METEOR increase when hidden state increases, but only slightly From ablation study, slight improvement but proposed CoDiSum performs better than variants
Experimental Setting	
Hardware	Not specified in the paper neither in the code repository
Software	From code repository: Python >= 3.6.2 Numpy >= 1.15.0 / Keras >= 2.1.5 / Tensorflow >= 1.7.0
Test Dataset	Test set used for testing is the same specified to be divided in dataset section. Checked from GitHub repository, in which the test is dealt with by setting indexes in which the dataset is divided, so same dataset is to be used.

General Information			
Name of experiment	CommitBERT	Authors	Tae-Hwan Jung
Year of publication	2021	Available online	Available at link https://github.com/graykode/commit-autosuggestions
Publication paper name	CommitBERT: Commit Message Generation Using Pre-Trained Programming Language Model	Notes on availability	Available and some important information not contained in the paper extracted from GitHub repository

Experiment Information

Comparison	
Treatments	Not compared to any other proposal on the field, only to different initialization of weights and biases and input type: CommitBERT v0.1(random),v0.2(RoBERTa),v0.3(CodeBERT) and v0 (original)(CodeBERT + Code-to-NL task)
Ablation study	Main comparison is an ablation study as it does not compare to external approaches, see Treatments section
Human evaluation	Not performed
Metrics	BLEU-4 and dev PPL (Perplexity)
Other variables	No other metrics used or statistical analysis performed
Training runs before reporting	Not mentioned. Mention run on ablation study with different initial parameters but not how many times before reporting
Comparison procedure	The comparison procedure consists on two main sections. First section takes another dataset from Loyola et al. With 26k data points and compares the results of the model using only added/deleted lines and the whole code diff as a unit The second part, resulting in the main conclusions of the paper, performs an ablation study on 345k dataset (main dataset) of the performance of the model with different weight initializations (see Treatments). Lastly, a case study is presented and analyzed for each language.
Experimental results	
Main results	CodeBERT + Code-to-NL is the best initialization to achieve highest metric values As future works, commitBERT should incorporate ASTs
Other results	From 2nd dataset used (Loyola et al.), using added/deleted lines is better than inputting the whole diff
Experimental Setting	
Hardware	V100 GPU, models have 16-32 GB of VRAM but not specified which. No mention of other hardware elements
Software	No mention in paper of software settings. From code we can extract: CUDA version 10 and CUDNN (support for AI development) version 7 sentencepiece==0.1.91 / transformers==3.4.0 / flask==1.1.2 / gitpython / jsonlines / tqdm / pydriller / transformers==3.4.0 / wandb / knockknock / pytorch
Test Dataset	Test set used for testing against treatments is extracted from main dataset for the approach Only in comparison of input type between whole modification or added/deleted lines dataset (not only test, but whole) is changed to Loyola et al, but concludes added/deleted is better and is not used in results against treatments

General Information

Name of experiment	COME	Authors	Yichen He, Liran Wang, Kaiyi Wang, Yupeng Zhang, Hang Zhang, Zhoujun Li
Year of publication	2023	Available online	Yes - at link https://github.com/hyc2026/come/tree/master
Publication paper name	COME: Commit Message Generation with Modification Embedding	Notes on availability	Link for intermediate models stored and final experiment

Experiment Information

Comparison	
Treatments	Treatments used to compare COME against are distributed in the different commit generation approaches: • Retrieval-based: NNGen version 3.1 and 3.2 and Lucene version 1.1 and 1.2 • Generation-based: CommitGen versions 1.1 and 1.2, CoDiSum version 0, CoreGen versions 1.1 and 1.2, and FIRA version 1 (only in CoDiSum dataset) • Hybrid Approaches: RACE versions 1.1 and 1.2, CoRec versions 1.1 and 1.2 • Pretrained Approaches: Code-T5-base version and CommitBERT versions 1.1 and 1.2
Ablation study	Ablation study performed on the proposal's embedding module. The aim is to check wether the modification embedding is increasing the performance of the system. To this end, three versions of the system are studied, being one the original approach (v0), and the other two versions 0.1 (Translation-- in paper)and 0.2 (Translation- in paper)of the approach, first removing both contextualization and modification embedding and second removing only the code contextualization. Three alternatives are evaluated on all datasets presented.
Human evaluation	Human evaluation conducted on 6 people. They do evaluate 100 random commits from CoDiSum dataset. Each participant evaluates 50 commits, each commit gets evaluated by 3 participants. The questionnaire design also incorporates shuffling the dataset in each question for the participants not knowing which one was generated by which approach. Commits are evaluated on relevance, usefulness and content adequacy.
Metrics	BLEU-4 - paper does not specify variation of BLEU but from code repository n = 4 ROUGE-L METEOR CIDEr - calculates cosine similarity of each sentence using TDF-IF vector at n-gram level
Other variables	Other statistical variables are used when analyzing the results of human evaluation (table in page798, or 7). For this, the authors use the average score, the variance, and the count of each score for each approach Also, Wilcoxon signed-rank test is performed and showed that human evaluation reliabylity is significant with a 98% confidence
Training runs before reporting	The training takes two runs before running the experiment.
Comparison procedure	The authors evaluate 4 aspects in the comparison.First, the comparison in all dataset for the test sets described is evaluated for each of the treatments proposed. After this, the results are commented and the better performance of some of the approaches in some specific dataset (better in MCMD than in CodiSum) is explained. Also, the human evaluation results are analyzed and the results of the ablation study discussed. Lastly, a case study on the results of COME is performed and, to further validate the effectiveness of the context module of COME, a JIT defect prediction test is performed, to see if the contextualization module can handle defect prediction using the context. For this last experiment, COME is compared to JIT, DNBJIT and DeepJIT using QT and OPENSTACK metrics, which are not related to commit message generation.

Experimental results	
Main results	COME outperforms baseline approaches by 9,2% on average of all evaluation metrics. (6,8% in BLEU, 8,7% on ROUGE, 11,7% on METEOR and a 9,5% on CIDEr) Modification embedding using fine-grained+simple structure seems to enhance performance of commit generation systems The separate modules from COME outperforms their respective competition in each category (retrieval and generation)
Other results	On the ablation study, conclusion shows that incorporation of both modules incresases a 53% and a 5,3% compared to only using retrieval or generation alone Dataset use shows that COME and retrieval approaches perform better in MCMD than in CoDiSum, apparently because of the similarity between training and test sets in MCMD, which does not occur on CoDiSum dataset On JIT defect detection, COME also outperforms treatments
Experimental Setting	
Hardware	Dell workstation with Intel Xeon Gold 6130 CPU @2.10GHz with 1 NVIDIA Tesla V100 32GB VRAM GPU
Software	Machine running OS Debian 5.4.143.bsk Installed libgcc-ng 9.1.0 / libffi 3.3 / pip 21.2.2 / python 3.6.13 / openssl 1.1.1s / readline 8.1.2 / setuptools 58.0.4 / sqlite 3.38.5 / tk 8.6.12 / wheel 0.37.1 / xz 5.2.5 / zlib 1.2.12 AI-related libraries installed by pip are huggingface-hub 0.4 / matplotlib = 3.3.4 / nltk 3.6.7 / numpy 1.19.5 / beautifulsopu4 4.11.1 / rouge 1.0.1 / scikit-learn 0.24.2 / scipy 1.5.4 / torch 1.10.0 / tensorboard 2.10.1 / tokenizer 0.10.3 (Rest of versions in environment.yml file in repository)
Test Dataset	The test sets used in the comparison of the paper are the same extracted in both datasets (MCMD and CoDiSum). In human evaluation only CoDiSum test set presented. In general comparison and ablation study all test sets used, MCMD distributed by programming language and CoDiSum

General Information			
Name of experiment	CoMeG	Authors	Shengbin Xu, Yuan Yao, Feng Xu, Tianxiao Gu and Hanghang Tong
Year of publication	2022	Available online	Available at link https://github.com/SoftWiser-group/CoMeG.
Publication paper name	Combining Code Context and Fine-grained Code Difference for Commit Message Generation	Notes on availability	N/A

Experiment Information	
Comparison	
Treatments	Treatments are said to be configured as CoMeG when they are Transformers and using default parameters when they are not: Neural Machine Translation - NMT versions 4.1 ,4.2 and 4.3 NNGen - not based in Transformer (assume default parameters as is no DNN) yet new dataset - versions 4.1, 4.2 and 4.3 CoDiSum versions 2.1, 2.2 and 2.3 ATOM versions 1.1, 1.2 and 1.3
Ablation study	Ablation study comparing four variations for the model. Variations 0.1 to 0.4 are: v0.1 = remove the diff modeling v0.2 = remove the context encoders and no pointer networks v0.3 = remove path embedding at input v0.4 = removes the AST diff encoder As for the four variations of the experiment, they are also trained in the three different datasets (partitioned differently), so each variation resolves in 3 sub-variations (0.X.1, 0.X.2 and 0.X.3) They are all evaluated on the same metrics as the main treatments
Human evaluation	No human evaluation performed
Metrics	BLEU-4 / ROUGE-L / METEOR
Other variables	No evidence on the paper of any additional metric used to statistically analyze the results.
Training runs before reporting	Not mentioned. From settings in proposal assumed 1 (because training steps = 1)
Comparison procedure	The comparison procecdure consists on measuring the treatments compared on the metrics over the three different partitions of the dataset presented. After it, ablation study is performed to study the gain of performance with the different mechanisms that modify the input format. Lastly, there is another experiment on how the modification of two parameters: hidden dimension and encoder/decoder layers, affect the performance of the metrics though it is not specified which is the dataset used for this part of the study. Lastly, a case study on the output of CoMeG and the rest of treatments is presented and judged by the authors.
Experimental results	
Main results	CoMeG outperforms treatments but not in every case (NNGen outperforms in random dataset due to potential similarities between training and test set) CoMeG performs in a consistent manner accross all partitions, not the same for other treatments Future works should consider partition by project as it might be an indicator of a threat to validity in experimetns
Other results	On the ablation study, CoMeG performs better with all the components except in random partition, in which CoMeG withou path embedding performs slightly better, so this component would have the least relevance on ablations Studies on the behavior of metrics when changing hidden size and layer number sugest that 128 and 3 are the optimals, though it is not presented in which dataset is this tested According to authors in case study, CoMeG seems to generate more natural messages than the other treatments
Experimental Setting	
Hardware	Server with 4 GPUs NVIDIA Tesla T4, they offer 16GB of VRAM each (not mentioned). No additional information mentioned.
Software	No mention on the paper. From code repository: nltk 3.7 / numpy 1.21.2 / pycocoevalcap 1.2 / torch 1.4.0 / tqdm 4.62.3 / transformers 4.6.0
Test Dataset	Test sets come from dataset explained in section dataset. As it is partitioned according to 3 criteria (randomly, timestamp and by project), 3 test sets resulted from these partitions. All 3 are used to extract the results in the comparison process.

General Information			
Name of experiment	CoRec	Authors	Haoye Wang,Xin Xia,David Lo,Qiang He,Xinyu Wang,John Grundy
Year of publication	2021	Available online	Available at https://zenodo.org/records/3828107
Publication paper name	Context-aware Retrieval-based Deep Commit Message Generation	Notes on availability	Link provided in paper (shorted by tiny.cc) not working, no mention of any other link in the paper for the proposal

Experiment Information	
Comparison	
Treatments	Other treatments to compare to: PtrGNCMsg version 2.1 and 2.2 (one for top 1,000 and other for top 10,000 dataset). Also version 2.3 split by project NNGen version 0 (+-0.01 precision same results) and version 5.1 (used on top 10,000 dataset). Also version 5.2 split by project For the main version of CoRec, there are two variants of the proposal (0.0.1 and 0.0.2), one for each dataset- Also version 0.5 split by project on top 10,000 dataset
Ablation study	Ablation study performed on the effectiveness of the various components that are included in CoRec. For this, authors create four variants of the experiments (0.1 to 0.4), being: v0.1 - with only the output of the retrieval module (without context-aware mechanism) v0.2 - basic model without decay sampling mechanism and no retrieval module v0.3 - not including the retrieval module v0.4 - not including the decay sampling mechanism As all the mentioned versions are trained in both datasets, there is a version of each ablation proposal for each dataset (0.X.1 and 0.X.2 respectively for each version). After showing the comparison of the different versions, statistical analisis is performed using Wilcoxon signed-rank test to test the statistical significance
Human evaluation	Human evaluation performed on 6 PhD students and 100 commits (50 from each dataset). Each commit group (a group is the 50 commits of one dataset) is evaluated by three participants. Examples in the questionnaire are shuffled for the participants not knowing which system does each come from. The commits are evaluated in Relevance, Usefulness and Content-adecuacy in a scale 1-5. When results from the human evaluation are recovered and analyzed, a Wilcoxon signed-rank test and a Bonferroni correction are performed in order to evaluate the statistical significance. Also, a Fleiss Kappa test is performed to measure the agreement between the responses of the participants. Apart from the human evaluation itself, another kind of human evaluation is conducted to test CoRec. The approach is tested by developers in a real-world scenario. From top 10,000 projects recovered, recent commits are recovered (between 2019/Dec/20 and 2020/Sep/08) and they are updated and sent back to the developers. Total of 2,656 commits from 1,154 developers collected. Participants are asked two yes/no questions about the satisfaction and helpfulness of the results and the tool, and a free-form question is asked about suggestions for the tool. Results are analyzed by the authors and conclusions extracted from that. Finally, another human evaluation is conducted on the gramatical correctness of the messages is performed using 3 persons with a score above 500 in CET6 English test. Each evaluate the same 200 randomly extracted commits on gramatical or ungramatical categories for the message
Metrics	BLEU-4, Modified N-gram Precision for N 1,2,3 and 4, ROUGE-L and METEOR
Other variables	Other metrics used for statistical analysis in the paper are: Wilcoxon signed-rank test to check significance on ablation study - 95% significance with deltas analysis Wilcoxon signed-rank test to check significance at a 99.9% confidence level for human evaluation Bonferroni correction to interpret the results from human evaluation Fleiss Kappa test to measure agreement between participants on human evaluation % Ratio increment on low frequency word use on both datasets between basic model and final CoRec Improvement % ratio on ablation study results Mean of relevnce, usefulness and content-adecuacy with p-value < 0.001 (for NNGen and PtrGNCMsg) on Human evaluation Box plotting analysis of results from human evaluation Percentual analysis on developer feedback on real-world scenario tests Average of messages with low gramatical correctness
Training runs before reporting	Mention of authors making 5 runs of the experiment for the main comparison with +-0.2% of BLEU variation on the results, which shows enough stability to report after 5 runs.
Comparison procedure	The comparison process is exhaustive and includes many little variations of the dataset. First, the authors run the datasets top 1,000 and top 10,000 in all the treatments mentioned above, presenting the results of the metrics for them. After it, ablation study on all the variations and both datasets mentioned is performed and analyzed, also significance of results via Wilcoxon signed-rank test on 95% confidence and analysis of deltas is showed. Once ablation study is finished, there is an analysis on the capabilities to generate low-frequency words (called Out Of Vocabulary in other papers). To this end, the increment ratio on basic model and final CoRec is analyzed when showing low frequency words. After it, authors retrieve low frequency commits and conform **two littel datassets of low frequency words** called low-freq-1,000 and low-freq-10,000. Metrics are then again checked for these low frequency datasets to check if the final CoRec has better capabilities to put low frequency words on the correct commits. After it, a study on the variation of the metrics when varying the parameters dimension size, number of layers and training batch size is performed to show the adjustment of these hyperparameters to the optimal value. Then, the human evaluation described is performed and results analyzed, followed by the developer feedback from the real-word scenario test. Finally, there are two variations of the study performed. On one side, first study analyzes how the removal of noisy data and VDO filter affects the performance. For this, **variations of both datasets without filtering noisy data and variation of top 10,000 without VDO filter** are used to measure the approaches and conclude. On the other hand, a study when **datasets are divided by project** is conducted and results are analyzed. Last, some case studies and examples are analyzed by the authors, illustrating the main limitations of the approach.

Experimental results	
Main results	When compared on both datasets against treatments, CoRec outperforms all the treatments with increases in +10% to +19% All the treatments perform better on top 10,000 dataset, indicating its value for future approaches and advantage of larger-scale Retrieval module performs variation from ablation study outperforms NNGen CoRec increases the capability to add low-frequency words from basic models CoRec is ranked highest from candidates in Human Evaluation, being the closer to reference message but still with some gap Fair agreement on the characteristics of the generated messages amongst the participants on human evaluation 68% of developers find tool results satisfactory and 73% finds tool useful. They ask to integrate it in IDEs Performance when trained in noisy data with less VDO filtering increases in comparison to filtered datasets Performance when split by project is still better in CoRec than other approaches compared to
Other results	Ablation study shows that both decay sampling and retrieval incorporation improved performance, and including both modules is still benefitial for the approach Incorporation of low-frequency words to results is much higher in CoRec than basic models and increases the lower the frequency Metrics are also better for CoRec when tested in dataset variations with low-frequency words Sensitivity analysis shows that 2 layers, and lower batch sizes are better. For dimension, around 1,000 cells is best performance but taking in consideration time to train it is decided to use 512 as the dimension 73% of developers would think that the tool is useful and they would use it in their daily development Noise-added variations seems to enhance the generalization capabilities of the approach When analyzing divided by project, the overall performance of all approaches decrease Gramatical correctness of messaged generated by CoRec is usually stable and, in average, only 3.0 and 2.0 (for each dataset respectively) are ungramatical messages
Experimental Setting	
Hardware	Server with 12 cores of a 3.6GHz CPU, 64GB of RAM memory and a GPU NVIDIA GTX 1080 with 8GB of VRAM
Software	Server running Ubuntu 16.04 with environment dependencies (extracted from code repository): python >= 3.5 / pytorch 0.4.1 / torchtext 0.3.1 / ConfigArgParse 0.15.2 / nltk 3.4.5 / numpy 1.14.2
Test Dataset	Test set used for main comparison is extracted from the division presented on both top 1,000 and top 10,000 datasets but variations of them are used for different parts of the experiment. For comparison and ablation study same test set For low frequency analysis test set from variation of only low-frequency word commits For noise and VDO filter analysis, tests set are extracted in same proportion but from different datasets since noisy commits are not filtered from both datasets and VDO filter is not passed to dataset top 10,000 For the test set of the division by project, it is different since the dataset is divided differently

General Information			
Name of experiment	CoreGen	Authors	Lun Yiu Nie, Cuiyun Gao, Zhicong Zhong, Wai Lam, Yang Liu, Zenglin Xu
Year of publication	2021	Available online	Available at link https://github.com/Flitternie/CoreGen
Publication paper name	CoreGen: Contextualized Code Representation Learning for Commit Message Generation	Notes on availability	Accesible at link, link provided in paper

Experiment Information	
Comparison	
Treatments	Variations from CoreGen are versions 0.2, 0.3 and 0.4 (trained in deduplicated dataset, with ICSM mechanism and with hybrid loss function) Treatments used to compare against CoreGen are: Neural Machine Translation - NMT versions 5 (main comparison, already presented in NNGen paper),6.1 (deduplicated dataset variation) and 6.2 (base NMT with CoreGen stage I mechanism implemented) NNGen versions 0.2 (same as in NNGen paper) and 6 (deduplicated dataset version) PtrGNCMsg version 3 (only trained in NNGen dataset) **BERT-fused version 1** - a new approach based in BERT adapted by the authors for the paper (not enough description of the approach)
Ablation study	Ablation study performed by removing the first stage of the training procedure. Aims to prove that the first training in specialized tasks, which in this case are learning code contextualization for explicit changes in code and implicit changes in binary files, are benefitial to performance of final model. Variation 0.1 performs stage II fine-tuning of parameters without initializiation using results from stage I, which basically consists on training a base Transformer. Results on the main metrics are presented and analyzed.
Human evaluation	No human evaluation performed.
Metrics	BLEU-4, ROUGE-1, ROUGE-2, ROUGE-L, METEOR
Other variables	Count and percentage of eliminated entries from original dataset when creating deduplicated variation of dataset
Training runs before reporting	No mention of number of runs of training procedure before reporting results
Comparison procedure	Comparison procedure first compares the main treatments and the ablation study version on the metrics for the experiment, conclusions are extracted for the performance of all approaches and ablation study objectives. Then, an analysis on the parameter sensitivity on mask rate, layer number and attention head size is presented, analyzing their effect on BLEU-4, ROUGE-L and METEOR . After that comparison, the entries removed from the original dataset used in the first comparison are presented, being duplicated entries that Liu et al. didn't notice on their cleansed dataset. After it, approaches (except PtrGNCMsg and ablation) are measured and compare again on this variation of the dataset, and also BERT pre-trained model is applied to commit generation and applied to the Transformer mechanism (BERT-fused approach) to compare it with CoreGen. Then, another experiment is performed on the removal of the two training stages, using only 1 training stage and changing the loss function for a combination of stages' loss functions. After it, to prove generalization of stage I, NMT base model is trained on results of stage I. Finally, to illustrate future research direction, authors implement an enhanced variation of CoreGen (achieves better results) using a In-statement Code Structure Modeling (ICSM) task to stage I of training.
Experimental results	
Main results	CoreGen outperforms all treatments in all metrics CoreGen also converges much faster than rest of treatments, at 25 epochs significative gain On deduplicated dataset all approaches received lower performance but CoreGen stills outperforms other treatments When compared to BERT-fused approach, CoreGen also obtain better results in metrics. When NMT is trained using the mechanism of stage I, performance of NMT increases significantly, showing usefulness of code contextualization learning tasks When ICSM task is applied in first stage of training, all metrics from original approach increase
Other results	Ablation study shows that training on code contextualization tasks is benefitial for CoreGen as it represents about half of the performance gain compared to rest of treatments When parameter sensitivity test is performed, taking in consideration computational cost and the trends when changing parameters, 2 layers with 6 attention heads and 0.5 mask rate are decided. When training is performed only in 1 stage with hybrid loss function, performance decreases, showing benefits of performing 2 stages of training not only in flexibility and adaptation of the procedure, but also in performance
Experimental Setting	
Hardware	No mention of hardware used in the paper.
Software	No mention of software used in the paper. From code repository, extracted: Conda environment running python 3.6.13 with pip 21.0.1 - created with miniconda3 cudatoolkit 10.2.89 / libgcc 9.1.0 / mkl 2020.2 / ncurses 6.2 / ninja 1.10.2 / numpy 1.19.2 / pytorch 1.5.0 with dependencies for CUDA 10.2 and CUDNN 7.6.5 / setuptools 52.0.0 / sqlite 3.35.4 / torchvision 0.6.0 / tk 8.6.10 / wheel 0.36.2 / zlib 1.2.11 Rest of verions in environment.yml file
Test Dataset	The test set for main comparison and rest of comparisons against CoreGen except deduplicated dataset results use the test set extracted from main dataset used in the paper In deduplicated comparison, deduplicated test set is used, containing a total of 301 less entries than original test set (not mentioned if division of deduplicated dataset is done again or just removal of entries from validation and test sets)

General Information			
Name of experiment	FIRA	Authors	Jinhao Dong, Yiling Lou, Qihao Zhu, Zeyu Sun, Zhilin Li,Wenjie Zhang, Dan Hao
Year of publication	2022	Available online	Available at link https://github.com/DJjjjhao/FIRA-ICSE
Publication paper name	FIRA: Fine-Grained Graph-Based Code Change Representation for Automated Commit Message Generation	Notes on availability	Link provided in paper, badges from ACM for experiment availability in paper

Experiment Information	
Comparison	
Treatments	Treatments used to compare FIRA against are: LogGen version 1 - reimplemented as code is said not to be available NNGen version 7 - Used in CoDiSum dataset not reimplemented CoreGen version 2 - Trained in CoDiSum dataset not reimplemented CoDiSum version 3 - reimplemented as code is said not to be available ATOM version 2 - trained in CoDiSum dataset not reimplemented CoRec version 2 - trained in CoDiSum dataset not reimplemented
Ablation study	Ablation study performed with the aim of checking the effectiveness of each component included on FIRA's proposal. To this end, 3 variations of FIRA are created (versions 0.1,0.2 and 0.3), being: v0.1 - Removing the edit operation inclusion in the graph from input preprocessing v0.2 - Degrading dual copy mechanism into single copy (no subtokens copied) v0.3 - Remove both components from previous ablations at the same time All the proposals are trained and evaluated on metrics except the evaluation performed on BLEU with penalty, in which ablation study variations are not included
Human evaluation	Human evaluation performed on 6 developers with experience coding in Java, the language of the dataset snippets, from 3 to 5 years. A questionnaire is set for the developers to evaluate the alternatives on a randomly selected set of 100 commits. Commits generated from each approach are not marked, so participants do not know which approach generated each option. Options are to be evaluated in a scale from 0 to 4. Approaches included on the human evaluation are FIRA, NNGen and CoDiSum. After extraction of results, Wilcoxon signed-rank test is performed on 95% level of confidence, showing results are statistically significant.
Metrics	B-Norm BLEU, Penalty variation of B-Norm BLEU, ROUGE-L and METEOR
Other variables	Other statistical techniques and metrics used in the paper include: Box plot analysis for ROUGE-L metric for all compared treatments Percentual gain on metrics over other treatments Count/frequency of subtoken copying Ratio of succesful copy of subtokens Percentage of scores by category and average score in Human Evaluation
Training runs before reporting	Not mentioned how many runs before reporting results with final configuration.
Comparison procedure	Comparison procedure starts by comparin the treatments presented over the used metrics, offering a comparison of metrics amongst them. After this, analysis on ROUGE-L metric is performed via a box plot, followed by the evaluation of the approaches on the penalty version of B-Norm BLEU metric. Then, the ablation study evaluates the alternatives of the approach created on metrics (except penalty BLEU) and analyzes the contributions of each module. When analyzing the contributions of subtoken copy processes, ablation proposal, main FIRA, NNGen and CoDiSum are compared on the subtoken copying capabilities. Finally, human evaluation procedure and results are presented, and a couple of case studies evaluated by the authors.
Experimental results	
Main results	FIRA outperforms all treatments, on BLEU by a range [7%,112%], on ROUGE-L by a range [9%,112%] and on METEOR by a range [16%,79%] Performance gains are bigger against retrieval proposals FIRA also outperforms other approaches on Penalty BLEU, even though all of them receive lower scores FIRA can correctly copy more subtokens and more infrequent subtokens than evaluated approaches on this cathegory (NNGen and CoDiSum) Commit messages generated by FIRA are considered as high-quality in human evaluation Also, in human evaluation, FIRA is the approach that generates highest ratio of high-quality messages over lowest ratio of low quality ones
Other results	Including edit operations increase performance and help model to capture fine-grained code changes, which was one of the main problems that the system aimed to solve Dual copy mechanism also boost performance and allow FIRA to outperform other approaches (NNGen and CoDiSum) on succesful subtoken copy ratio
Experimental Setting	
Hardware	Dell workstation with Intel Xeon CPU E-2680 v4 @ 2.4GHz with 2 24 GB VRAM NVIDIA RTX 3090 and 2 24GB VRAM TITAN RTX
Software	Workstation running Ubuntu 16.04.6 LTS with Python 3.8.5 Pytorch 1.7.1 / Numpy 1.19.2 / Scipy 1.5.4 / Nltk 3.5 / Sacrebleu 1.5.1 / Sumeval 0.2.2
Test Dataset	Test set used to measure and compare with other treatments, as well as perform the ablation study and the rest of comparison procedures is the same test set that comes from dataset division.

General Information			
Name of experiment	RACE	Authors	Ensheng Shi, Yanlin Wang, Wei Tao, Lun Du, Hongyu Zhang, Shi Han, Dongmei Zhang, Hongbin Sun
Year of publication	2022	Available online	Available at https://github.com/DeepSoftwareAnalytics/RACE
Publication paper name	RACE: Retrieval-Augmented Commit Message Generation	Notes on availability	Code accesible and link provided in paper

Experiment Information	
Comparison	
Treatments	Treatments used to compare against RACE are: Lucene version 1.1 NNGen version 3.1 NMT (named CommitGen in paper, confusion) version 4.1 CoDiSum version 4 NMTGen (Loyola) versions 2.1 and 2.2 PtrGNCMsg version 4 ATOM version 3 (only on Java subset due to configuration) CoRec version 1.1 CommitBERT versions 1.1 and 2 CodeT5-small versions 1.1 and 1.2 CodeT5-base versions 1.1 and 2
Ablation study	Ablation study performed on RACE (version 0.1) removing the exemplar guider from Generation module and testing in all metrics as rest of treatments. It aims to demonstrate efectiveness of exemplar guider component in overall RACE approach.
Human evaluation	Human evaluation performed on 4 volunteers with English and programming skills. NNGen, NMTGen (Loyola), CommitBERT, CoRec and RACE compared in human evaluation. 50 random commits from each subset of MCMD dataset (for the five languages) extracted and evaluated by the volunteers. Each volunteer scores commit from 0 to 4 in three cathegories: Informativeness, Conciseness and Expresiveness. Final score for selected commit is scored based on the average score. To verify the agreement, statistical tests are performed (Krippendorff's Alpha and Kendalls' Tau), and to verify the significance of the results, Wilcoxon signed-rank test is performed on 95% confidence level
Metrics	B-Norm BLEU, ROUGE-L, METEOR and CIDEr
Other variables	Other statistical analysis performed include; Percentual increase for metrics from RACE to other approaches average Percentual gain of 4 models with RACE approach Average score from human evaluation Standard deviation of human evaluation scores Krippendorff's Alpha, Kendalls' Tau and Wilcoxon signed-rank test on 95% confidence for human evaluation
Training runs before reporting	3 runs with seed = 0/1/2 made and average scores for three run reported as results
Comparison procedure	Comparison procedure starts by presenting a table comparing all 11 treatments and the ablation study variation, with presented metrics measured in each of the subsets of the dataset divided by programming language. After it, a study is performed on using different k (number of recalled metrics from training corpus in Retrieval module. Then, to prove the value of the proposal, diff encoders trained for RACE approach are applied to Seq2Seq proposals (NMTGen v1.2, CommitBERT v2, CodeT5-small v1.2 and CodeT5-base v1.2) and gains in BLEU metric are measured. Finally, human evaluation is performed and results are discussed
Experimental results	
Main results	RACE outperforms all the rest of treatments and the ablated version in all four metrics evaluated, gains from 6% to 46% All models enhance performance over the original models of NMTGen, CommitBERT, CodeT5-small and CodeT5-base on 7% to 73% when applying pretrained components from RACE High degree of agreement and significance from significance and agreement statistical tests in human evaluation concludes RACE is better in three aspects measured in human evaluation
Other results	Ablation study concludes that exampler guider is an important component of RACE which enhances performance Amongst different number of recalled entries in Retrieval mode, performance is stable, indicating consistency of approach
Experimental Setting	
Hardware	Server with 4 GPUs NVIDIA Tesla V100. No more hardware mentioned
Software	No mention in paper of software versions. From code repository extracted Conda environment with Python 3.6 torch 1.10 / transformers 4.12.5 / tdqm 4.64.1 / prettytable 2.5.0 / gdown 4.5.1 / more-itertools 8.14.0 / tensorboardX 2.5.1 / setuptools 59.5.0 / tensorboard 2.10.1
Test Dataset	Test set used to run experiments, to ablation study and to extract commits for human evaluation is the same test set extracted from dataset in dataset section

General Information			
Name of experiment	ReGenSD	Authors	Zhihan Li, Yi Cheng, Haiyang Yang, Li Kuang, Lingyan Zhang
Year of publication	2022	Available online	Not available at link provided https://anonymous.4open.science/r/regensd-61E5/
Publication paper name	Retrieve-Guided Commit Message Generation with Semantic Similarity And Disparity	Notes on availability	Anonymous 4open science site shows expired repository when trying to access to the source code of the experiment

Experiment Information	
Comparison	
Treatments	Treatments to compare agains ReGenSD are: NMT version 6 PtrGNCMsg version 2.1
Ablation study	Ablation study performed on several levels. First ablation study aims to check the relevance of also taking the message as part of the input in the retrieval-based generation. This ablation is version 0.1 Second ablation study experiment performed present variations of the mechanisms used in Encoder and Decoder of Guide module, aiming to study the positive aspects brought by these mechanisms. Paper presents variations: v0.2 - No additional mechanisms in RNN Encoder-Decoder v0.3 - Only mechanism is Selection module applied v0.4 - Only mechanism is Relation Gate used in Encoder v0.5 - Only mechanism is Difference Vector in Decoder v0.6 - Removes Relation Gate from original appoach v0.7 - Removes Difference Vectors from original approach v0.8 - Removes Selection module from original approach All variations are measured on same test set and same metrics, analyzing after the analysis the results
Human evaluation	No human evaluation performed
Metrics	BLEU (avg), BLEU-1, BLEU-2, BLEU-3 and BLEU-4
Other variables	No other relevant statistical analysis performed over the presented resutls
Training runs before reporting	Paper mentions "many runs" of the proposal, taking the average as final results, which contain a variation on metrics of around 0.25%
Comparison procedure	Comparison process starts by comparing the original approach to both treatments over the specified metrics. After it, ablation study on the use of retrieved messages associated to the entry performed, follow by the ablation on the use of different mechanisms to the Encoder-Decoder of Guide module. Results are analyzed and variations 0.2, 0.3 and 0.8 are compared to treatment NNGen to study which components add negative information to the output of retrieval systems. Last, a case study is presented by the authors on the results of ReGenSD and other treatments
Experimental results	
Main results	ReGenSD outperforms rest of treatments, from 60% to 103% against NMT and from 16% to 60% against PtrGNCMsg All methods using retrieval results without using selective mechanisms, including NNGen, perform worse than ReGenSD with the selection module
Other results	Almost all evaluation metrics (except BLEU-1) enhance by using the paired messages from retrieved entries rather than only the code diff Complete ReGenSD model performs better than all ablations and removing any mechanism decreases performance All models with one variation only perform better than ablation without any variation
Experimental Setting	
Hardware	Computer with NVIDIA GeForce RTX 2080 Ti with 11 GB of VRAM. Not further mention of hardware
Software	Running on Ubuntu 16.04. Uses Tensorflow framerowk. No additional information can be extracted since repository is not accesible
Test Dataset	Test set used is same test set extracted in dataset division section from NNGen Liu et al. Cleansed dataset, it is used for all comparisons and in metrics measurement in paper

General Information			
Name of experiment	CommitGen	Authors	Pablo Loyola, Edison Marrese-Taylor and Yutaka Matsuo
Year of publication	2017	Available online	Available at link https://github.com/epochx/commitgen
Publication paper name	A Neural Architecture for Generating Natural Language Descriptions from Source Code Changes	Notes on availability	Experiment available at link, link provided in paper

Experiment Information	
Comparison	
Treatments	reatment used to compare against CommitGen is MOSES version, only in atomic dataset. MOSES is not a DNN approac
Ablation study	Ablation study not performed. Nevertheless, different variation of dataset used to train twice CommitGen. First being trained on atomic dataset to compare against MOSES and then trained in full dataset (version 0.1) to measure metrics
Human evaluation	Not performed
Metrics	METEOR (named Validation Accuracy), BLEU-4
	No other statistical analysis performed on the paper.
Training runs before reporting	Not mention of how many runs before reporting results
Comparison procedure	Comparison process is based on training in atomic dataset both approaches and compare their METEOR (Val. Accuracy) and BLEU metrics. Also, model is trained on full dataset to measure its performance on BLEU metric. A table of generated and reference messages for CommitGen is showed, as well as a heat map of the attention weights.
Experimental results	
Main results	The idea of generating commit messages using DNN models is feasible when results compared to approaches like MOSES Model was able to learn representation shows that model learns better than MOSES in consistency and representations with compressive power
Other results	Performance in Java is bigger and authors hypothesize is because of changes are expected to be longer in Java Model is said to be able to generate semantically sound description according to case studies Model has tendency to choose more general terms over specific ones
Experimental Setting	
Hardware	No mention of harware used in paper
Software	No mention of software used on paper. On code repository, mentioned needed packages are Torch, Cutorch, unidiff and pygments. No version mentioned
Test Dataset	No possibility of establishing if test set is same as no division of dataset is explicitly mentioned in paper Assume there are two different test sets since models' results are reported on both full and atomic versions of dataset

General Information			
Name of experiment	Mucha	Authors	Chuangwei Wang, Yifan Wu, and Xiaofang Zhang
Year of publication	2023	Available online	Link provided https://github.com/cmgads/Mucha - Not working
Publication paper name	Mucha: Multi-channel based Code Change Representation Learning for Commit Message Generation	Notes on availability	Link provided show error 404 not found when trying to access. Not possible to find repository from online search

Experiment Information

Comparison	
Treatments	Treatments used to compare Mucha against are: CoDiSum version 5 CoreGen version 3 CCRep version 1 (not specific for commit message generation) CodeBERT version 1 (not specific for commit message generation as variant CommitBERT) UniXcoder version 1 (for general code task, not specifically commits)
Ablation study	Ablation study performed on the three channels that correspond to each level of the fine-grained representation (versions 0.1, 0.2 and 0.3). Also, ablation study performed on the initialization of the pre-trained model using CodeBERT (version 0.4) parameters instead of UniXcoder. Ablation study performed on same metrics as comparison against treatments.
Human evaluation	No human evaluation performed
Metrics	BLEU-4, ROUGE-L, METEOR
Other variables	No other statistical variables or analysis of the results performed further than the expression in percentual points of the results of the main metrics
Training runs before reporting	All experiments were run 10 times to ensure stability and fairness of the experiments
Comparison procedure	The comparison procedure starts by training 10 times each of the treatments that have been adapted or used to compare against Mucha. First, results on the three main metrics are presented. After this, results are commented and special cases when Mucha is outperformed (ROUGE-L) are explained. Then, ablation study on multi-channels and initialization of weigths and biases is presented and results are discussed. Finally, an analysis on the trends and performance of the approach when changing the parameters input size and initial learning rate for the Adam optimizer is studies and explained
Experimental results	
Main results	Mucha outperforms all treatments except UniXcoder in ROUGE-L. Improvements of at least 18.2% (BLEU), 72.2% (METEOR), and 10.5% (ROUGE-L, except UniXcoder) Results on three metrics for all approaches are still very low
Other results	Model performance degrades when removing channels from model, drop is the most significant when AST channel is removed General trend is to have better performance with higher input size, length set at 512 because of hardware limitations
Experimental Setting	
Hardware	Server with 24 cores at 3.8 GHz CPU and a NVIDIA RTX 3090 GPU
Software	HuggingFace's Transformer python package used and PyTorch deep learning framework used. No further mention of software used. Since repository is not accesible it is not possible to extract more specifications for software
Test Dataset	It is said in paper that all experiments are conducted in their dataset, so it is feasible to assume that test set used is the same that was created from dataset division. Nevertheless, dataset division is not mentioned on the paper, so no additional details on the test set used in experiments since code and dataset are not available.

General Information			
Name of experiment	CCT5	Authors	Bo Lin, Shangwen Wang, Zhongxin Liu, Yepang Liu, Xin Xia, Xiaoguang Mao
Year of publication	2023	Available online	Available at link https://github.com/Ringbo/CCT5
Publication paper name	CCT5: A Code-Change-Oriented Pre-trained Model	Notes on availability	Code available, link provided in paper

Experiment Information	
Comparison	
Treatments	Treatments used to compare against CCT5 (in commit message generation) are: NNGen version 8 CodeT5 version 3 CodeReviewer version 1 FIRA version 2 (Treatment reported but no results of comparison showed)
Ablation study	Ablation study performed on the removal of pre-training tasks to check performance variations, but variations only checked in Just-in-time-defect-detection task, not entirely related to commit message generation capabilities.Therefore, variations not presented as fall out of commit message generation scope
Human evaluation	Human evaluation performed on the commit message generation task. For human evaluation, 5 PhD. students participate, evaluation 100 entries and results (20 for each language in the dataset) and they are fed to four approaches (excluding FIRA as cannot accept various languages). Results are evaluated on a scale 1 to 5 on three aspects: Adequacy, Conciseness and Expresiveness. Four approaches are anonymous in the questionnaire to avoid bias.
Metrics	Metric used to measure commit message generation is B-Norm BLEU (it is said to have highest correlation with human scores)
ariables	Average results of approaches in human evaluaion Wilcoxon signed-ranked test perform on significance of results for metric results
Training runs before reporting	No mention of training times before reporting final results
Comparison procedure	Comparison procedure presented in paper does not only cover commit message generation tasks, but also just-in-time defect detection and just-in-time comment update. As two of them fall out of the scope of commit message generation, their comparison procedures are not covered here. For commit message generation, approaches are trained in MCMD dataset and compared over its test set. As for FIRA, it is said that results are taken from Dong et al. as the MCMD dataset contains multiple languages and FIRA only accepts Java. Despite this, table showing results of FIRA approach is not presented in the paper. After this, human evaluation is presented on the commit message generation task, and a case study is also present to illustrate CCT5's capabilities on generating commit mesages
Experimental results	
Main results	CCT5 consistently outperforms other treatments with an average B-Norm BLEU of 22.06% CCT5 outperforms by a bigger gap non pre-trained-based approaches (NNGen) Despite FIRA results not being showed, it is said that CCT5 outperforms FIRA by a 9.2%
Other results	Human evaluation shows that differences are significant on a 95% confidence level
Experimental Setting	
Hardware	Server with 2x NVIDIA GeForce 4090 GPUs
Software	Paper only states popular framework PyTorch From code repository, extracted: Conda environment with Python version 3.8 pytorch=2.0.0 / torchvision=0.15.1 / torchaudio / datasets==1.16.1 / transformers==4.21.1/ tensorboard==2.12.2 / tree-sitter==0.19.1 / nltk=3.8.1 / scipy=1.10.1
Test Dataset	Test set used in evaluation is not the same as in model pre-training and fine-tuning For approaches except FIRA, test set comes from 80-10-10 split of MCMD dataset For FIRA (results not reported), evaluation is performed on test set coming from CoDiSum dataset

General Information			
Name of experiment	N/A	Authors	Wei Tao,Yanlin Wang, Ensheng Shi, Lun Du, Shi Han, Hongyu Zhang, Dongmei Zhang and Wenqiang Zhang
Year of publication	2022	Available online	Available at link https://anonymous.4open.science/r/CommitMessageEmpirical
Publication paper name	A large-scale empirical study of commit message generation: models, datasets and evaluation	Notes on availability	Experimental review and results available, link provided in paper

Experiment Information	
Comparison	
Treatments	Treatments analyzed in the empirical study are: CmtGen versions 1.1 to 1.6 and 1.7 (ablation on noisy data) NMT versions 7.1 to 7.6 NNGen versions 9.1 to 9.6 CoRec versions 3.1 to 3.6 PtrGNCMsg versions 5.1 to 5.6 CoDiSum versions 6.1 to 6.6 ATOM version 4 (only in MCMD Java version) Versions X.1 to X.8 correspond to X.1-Trained in CmtGen dataset X.2-Trained in NNGen dataset X.3-Trained in CoDiSum dataset X.4-Trained in MCMD dataset X.5-Average of training in MCMD split by project X.6-Average of training in MCMD split by timestamp
Ablation study	Ablation study perform on CmtGen approach to study the effect of noisy data on the dataset. To this end, CmtGen dataset is completed with noisy commits and performance of CmtGen approach is tested both in original dataset and noise-added one
Human evaluation	Human evaluation is peformed on 100 randomly selected commits with 3 BLEU variants over 30 in value, to asses which of the variations of the BLEU metric correlates best with human evaluation of generated outputs. Selected commits are evaluated on three main aspects: Content adequacy, Conciseness and Expresiveness, in a scale from 0 to 4. To validate the results, authors perform a Krippendorf's alpha and Kendall's Tau tests to see the agreement degree between participants. Later, correlation between results and human evaluation scores are compared to see which variation of BLEU correlates the best
Metrics	Metrics used are B-Moses (BLEU implemented for MOSES), B-CC (BLEU from NLTK with smoothing method), B-Norm BLEU, ROUGE-1, ROUGE-2, ROUGE-L and METEOR
Other variables	Other statistical procedures performed are: Krippendorf's alpha and Kendall's Tau on human evaluation Retrieval index for retrieval approaches Percentual improvement on timestamp division Average inference time for MCMD
Training runs before reporting	No mention of number of training runs before reporting results about the treatments used in the empirical study
Comparison procedure	Comparison procedure in the empirical study starts with measuring the performance of selected approaches in different existent datasets (all Java based) and comparin against the Java section of presented MCMD dataset. After it, correlation between human evaluation and variations of BLEU is presented according to human evaluation procedures. Then, ablation study on CmtGen is presented by the explained methodology, followed by the variation of performance of selected approaches with different lengths intervals of the test set. Also, the inference time of each approach in all different PL sections of MCMD datasets is presented. Finally, tables containing measures for all metrics in MCMD dataset and subdivisions is presented, as well as scores for timestamp and project split of MCMD dataset and its subsections. CodeBERT is also studied in various steps of the empirical review, tohugh, as it is not specifically aimed for commit message generation, it is not included in results
Experimental results	
Main results	Results are not consistent for one approach in different metrics and datasets, but overall, CoRec and NNGen perform best in MCMD dataset, while when splitting the dataset by different criteria, PtrGNCMsg perform best Most datasets are Java-based, performance in MCMD (being not only Java-based) reduces perfomance measured because many approaches are Java oriented in their design
Other results	Introducing noise in CmtGen dataset boosts performance in B-Norm approach for approach. Nevertheless, generating the first verb correctly has a great effect in BLEU metric
Experimental Setting	
Hardware	Two Tesla V100 GPUs
Software	No explicit description of software used to perform the experiments in the paper From code repository: Conda environment with Python 3.8 numpy 1.19.2 / nltk 3.6.2 / scipy 1.5.2 / pandas 1.1.3 / krippendorff 0.4.0 / scikit-learn 0.24.1 / sumeval 0.2.2 / sacrebleu 1.5.1 / matplotlib 3.5.1
Test Dataset	Testing sets come from original division of each of the used datasets, except in ablation study of CmtGen dataset, in which noisy entries are added to the test set to check performance with noise

Dataset 1: CmtGen	
Dataset Origin	Preprocessed dataset first proposed in Jiang and McMillan for generation of a verb from the commit data, exploring top 1,000 Java projects on GitHub order by star number and cleansed/filtered by Liu et al. 2018. Original dataset contained ~1.6M commits
Dataset Composition	Content of dataset is around 32,000 preprocessed pairs of <diff,message> extracted from top 1,000 java projects in order of stars, extracted from 1.6M+ commits from Jiang and McMillan dataset
Dataset Adjustment/Processing	Take datapoint from reference dataset of Jiang and McMillan Filter first sentence, remove commit ID, issue ID, merge and rollback commits Remove commits larger than 1Mb, filter out commits longer than 30 words (msg) or 100 tokens (diff) Apply a VDO (Verb-Direct Object) filter - using Standford CoreNLP
Dataset Division	Random division of dataset into: Training set - 26,000 pairs (81.25%) Validation set - 3,000 pairs (9.375%) Test set - 3,000 pairs (9.375%)
Dataset available online	Dataset available in online annex at link https://drive.google.com/drive/folders/1HSQ2HWfIzf886Zh4X9NPAFKV-indpjZ7
Dataset 2: NNGen	
Dataset Origin	Comes from cleaning the dataset used in the Original proposal for NMT. Explained above. It is used by the authors to retrain NMT and then to compare with NNGen approach.
Dataset Composition	Cleansed 27,144 pairs of <diff,message> from top 1,000 Java projects in order of descending stars.
Dataset Adjustment/Processing	Take original dataset from NMT Detect bot-generated messages according to liferay-continuous-integration pattern - around 13% in each set Detect trivial messages according to trivial patterns proposed by the authors - around 3% in each set Remove detected messages from original dataset
Dataset Division	Dataset divided randomly in: Training set - 22,112- (81,40%) Validation set - 2,511 - (9,25%) Test set - 2,521- (9,35%)
Dataset available online	Dataset available in online annex at link https://drive.google.com/drive/folders/1HSQ2HWfIzf886Zh4X9NPAFKV-indpjZ7
Dataset 3: CoDiSum	
Dataset Origin	Dataset used for CoDiSum comes from 509k+ Jiand and McMillan dataset preprocessed and filtered to enchance quality of resulting dataset. Comes from top 1,000 Java projects ordered by stars
Dataset Composition	The dataset, after preprocessing, contains 90,661 pairs of <diff, message> extracted from preprocessing Jiang and McMillan dataset.
Dataset Adjustment/Processing	From Jiang and McMillan dataset: Remove non .java file related commits and remove code of files with no code diffs Use NLP to remove punctuation and special symbols Tokenize diffs and message and remove < 3 words Delete duplicate commits and various modifications in short period of time
Dataset Division	Random division of pairs from dataset into: Training set - 75,000 pairs - (82.7%) Validation set - 8,000 pairs - (8.8%)
Dataset available online	Dataset available at same link for experiment - https://github.com/SoftWiser-group/CoDiSum
Dataset 4 MCMD	
Dataset Origin	Large dataset called MCMD proposed by Tao et al. ICSME '21.
Dataset Composition	Contains filtered pairs <diff,message> from dataset proposed in mentioned paper. It contains entries in 5 programming languages (C++, C#, Java, Python and JavaScript) from top 100 projects on GitHub. For each language: C++ - ~200,000 entries C# - ~188,000 entries Java - ~200,000 entries Python - ~257,000 entries JavaSript - ~250,000 entries
Dataset Adjustment/Processing	Original dataset contains entries for each of the 5 programming languages Filter redundant messages such as merge and rollback commits Filter out noisy message as stated in Liu et al. 2018 To balance size data, they retain 450,000 commits for each programming language Authors filter out commits with multiple files and non-parseable files (.mp3,.jar...)
Dataset Division	Dataset is divided in the 5 programming languages (C++/C#/Java/Python/JavaScript): Training set - 160,948/149,907/ 160,018/206,777/ 197,529 - Total 875,179 (~80%) Validation set - 20,000/18,688/19,825/25,912/24,899 - Total 109,324 (~10%) Test set - 20,141/18,702/20,159/25,837/24,773 - Total 109,612 (~10%)
Dataset available online	Available at link https://doi.org/10.5281/zenodo.5025758

General Information			
Name of experiment	N/A	Authors	Jinhao Dong, Yiling Lou, Dan Hao, Lin Tan
Year of publication	2023	Available online	Available at link https://zenodo.org/records/7042270
Publication paper name	Revisiting Learning-based Commit Message Generation	Notes on availability	Files available, link provided in paper

Experiment Information	
Comparison	
Treatments	Treatments presented in empirical study are: NMT versions 8.1 to 8.3 PtrGNCMsg versions 6.1 to 6.4 CoDiSum versions 7.1 to 7.5 FIRA versions 3.1, 3.2, 3.4 and 3.5 (No 3.3, no attention mechanism ablation) CoreGen versions 4.1 and 4.2 Where version X.1-Version scored in training set and test set X.2-Version scored on only marks input X.3-Version with attention ablated X.4-Version with copy mechanism ablated X.5-Version with anonymization ablated
Ablation study	Ablation study performed to all approaches in different modules (main ablation in paper). The study explores the effectivity of each of three general modules: attention mechanism, copy mechanism and anonymization mechanism. For all approaches, ablations are performed if approaches incorporate the ablated mechanism. For NMT, ablation performed on attention mechanism, for PtrGNCMsg, ablation performed on attention and copy mechanisms, for FIRA, ablation performed on attention and anonymization mechanisms, and for CoDiSum ablation performed on all mechanisms. CoreGen not presented in ablation study. Reported results of ablation study do not only report metrics for ablation, but also from generated messages following patterns and not following patterns, since is in the scope of the study
Human evaluation	No human evaluation performed in the paper
Metrics	Metrics used are BLEU (no ngram specified, according to results and citations, assumed BLEU-4) ,ROUGE-L and METEOR
Other variables	Other statistical variables considered in the study are: Pattern ratio, measuring the proportion of the message belonging to a certain message pattern Percentual increase/decrease on metrics between pattern messages and no-pattern messages
Training runs before reporting	Mentioned that dataset is divided using random criteria 5 times using same ratio and results are the average of them since average std. deviation is less than 0.30. Nevertheless, it is not explicitly mentioned if it is for dataset splitting purposes or if treatments are trained in all 5 divisions
Comparison procedure	Comparison procedure in paper starts by measuring the treatments on the designed metrics over both the training set of the used dataset and the test set. After it, it present the results of recognizing patterns over the generated messages, followed by a case study illustrating the patterns. After it, influence of the length of the commit message over the appearance of patterns in messages is studied. Then, treatments' scores are presented divided in pattern or no-pattern messages. Next section studies the main influence of the dataset, by changing the input of the treatments from the original dataset entries to only the code marks related to commits. Then, performance on both marks and original entries is divided into pattern an no-pattern groups. Finally, ablation study on components mentioned is presented, scored by the metrics used in the study, followed by an attention weight analysis and another analysis on most frequent tokens as input to each treatment
Experimental results	
Main results	All treatments analyze only on code marks, when generation-based, perform relatively stable In the study, in both training and test sets, FIRA usually outperforms rest of treatments
Other results	Around 90% of generated commit messages follow a pattern, significant increase compared to ground-truth (50%) High pattern ratio in training set influences/relates as high pattern ratios in generated messages
Experimental Setting	
Hardware	No mention of hardware used for tests
Software	No mention of software used for tests
Test Dataset	For each section, dataset changes, as for main comparison, results are extracted from test set presented on the dataset, training set is also used to evaluate the performance. Not only this, but dataset split is performed in 5 different manners and the presented results are the average of all 5 (explained on trainining runs before reporting). So, while the data for the results come from the same dataset, many different tests sets are applied to obtain the results.

Dataset	
Dataset Origin	Dataset used for CoDiSum comes from 509k+ Jiand and McMillan dataset preprocessed and filtered to enchance quality of resulting dataset. Comes from top 1,000 Java projects ordered by stars
Dataset Composition	The dataset, after preprocessing, contains 90,661 pairs of <diff, message> extracted from preprocessing Jiang and McMillan dataset.
Dataset Adjustment/Processing	From Jiang and McMillan dataset: Remove non .java file related commits and remove code of files with no code diffs Use NLP to remove punctuation and special symbols Tokenize diffs and message and remove < 3 words Delete duplicate commits and various modifications in short period of time
Dataset Division	Random division of pairs from dataset into: Training set - 72,529 pairs - (80%) Validation set - 9,060 pairs - (10%) Test set - 9,061 pairs - (10%) **Dataset split perfomed 5 times randomly, obtaining 5 variations of dataset**
Dataset available online	Dataset available at same link for empirical study - https://zenodo.org/records/7042270

General Information			
Name of experiment	N/A	Authors	Sven van Hal, Mathieu Post and Kasper Wendel
Year of publication	2019	Available online	Not available from information in paper
Publication paper name	Generating Commit Messages from Git Diffs	Notes on availability	No link provided for created dataset and variation of NMT

Experiment Information	
Comparison	
Treatments	Treatment used in study is Neural Machine Translation (NMT), which is reproduced and slightly changed to enhance its performance. Versions of NMT in this paper are: Version 8.1 (trained on Java dataset proposed in this paper) Version 8.2 (trained on C# dataset proposed in this paper) Version 8.3 (trained on original dataset but slightly different results) Version 8.4 (trained on processed version of original dataset)
Ablation study	No ablation study performed in paper
Human evaluation	No human evaluation performed in paper
Metrics	BLEU (assumed BLEU-4) , ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-W
Other variables	No additional statistical analysis performed on the experimental results
Training runs before reporting	No mention of number of runs before reporting results in paper
Comparison procedure	Comparison procedure consists on, after presenting the slight variations performed on hyperparameters and training set of NMT, extract the scores on used metrics for all datasets used in paper, followed by a discussion on the results of the reproducibility and influence of different datasets and performance
Experimental results	
Main results	Results from Jiang et al. can be reproduced achieving slighly better results on metrics on same dataset Preprocessing and input change did not improve results on metrics over original NMT approach
Other results	Performance on presented datasets are better than processed version of Jiang et al. Performance of NMT said to be dependant on memorizing long paths
Experimental Setting	
Hardware	NVIDIA GTX 1660 GPU with 6GB of VRAM
Software	No mention of software used in paper, unavailability of code repository does not allow to extract more details
Test Dataset	Test sets used to report results are the resultant test sets for the presented datasets in the paper, divided as said or referenced in the dataset section

Dataset 1: Java and C# Datasets	
Dataset Origin	Dataset collected by the authors similar to Jiang et al. work but incluing not only Java projects, but C# projects as well from top 1,000 projects in GitHub
Dataset Composition	The dataset is split in two datasets, one for each language: Around 151,000 pairs <diff, message> from top 1,000 Java projects in GitHub ordered by stars Around 389,000 pairs <diff, message> from top 1,000 C# projects in GitHub oredered by stars
Dataset Adjustment/Processing	Both datasets are preprocessed the same: GitHub API is used to collect commit history from top 1,000 projects in GitHub, both in Java and C# Most recent commits for each branch are retrieved and the message and raw diff are kept Merge and initial commits, as well as diffs bigger than 1MB, are ignored Messages encoded in UTF-8 or corresponding unicode replacement character After it, same preprocessing than Jian et al. dataset is used, consisting in: For messages: Version and issue numbers replaced by placeholders, non-English characters removed and message lowercased Tokenized using NLTK Punkt sentence tokenizer. Messages with less than 2 or more than 30 tokens are removed, and VDO filter is still applied in a more relaxed criteria than Liu et al. For diffs: They are split in blocks, location of change removed and kept filename, extension and context of change Subtokens split, non-English characters removed and all lowercased Diffs with more than 100 tokens are discarded and remaining tokenized using Word-PuntctTokenizer
Dataset Division	Dataset randomly divided using only 360,000 entries for each language (then in second preprocessing this might be reduced): Training set - 288,000 - (80%) Validation set - 36,000 - (10%) Test set - 36,000 - (10%) Numbers provided for each dataset, as they are divided in Java and C# datasets
Dataset available online	Dataset link not provided
Dataset 2: NMT Jiang et al.	
Dataset Origin	Dataset used is a preprocessing adaptation from 2M Dataset proposed by Jiang and McMillan.
Dataset Composition	Content of dataset is 32,000 preprocessed pairs of <diff,message> extracted from top 1,000 java projects in order of stars, extracted from 2M+ commits from Jiang and McMillan dataset
Dataset Adjustment/Processing	Take datapoint from reference dataset of Jiang and McMillan Filter first sentence, remove commit ID, issue ID, merge and rollback commits Remove commits larger than 1Mb, filter out commits longer than 30 words (msg) or 100 tokens (diff) Apply a VDO (Verb-Direct Object) filter - using Standford CoreNLP
Dataset Division	Random division of dataset into: Training set - 26,000 pairs (81.25%) Validation set - 3,000 pairs (9.375%) Test set - 3,000 pairs (9.375%)
Dataset available online	Not available, said it is with link https://sjiang1.github.io/commitgen - Not accesible, new link not provided in this paper

Dataset 3: Jiang et al. Processed	
Dataset Origin	Dataset is a preprocessed version of previously presented dataset, made by the authors for evaluating purposes over NMT reproduction as preprocessing by Jiang et al. can be improved, according to the authors
Dataset Composition	Dataset contains around 156,000 pairs of <diff,message>. They are, again, extracted from Jiang et al. dataset, whose origin is still at Jian and McMillan top 1,000 Java projects on GitHub by descending star order
Dataset Adjustment/Processing	For messages: Version and issue numbers replaced by placeholders, non-English characters removed and message lowercased Tokenized using NLTK Punkt sentence tokenizer. Messages with less than 2 or more than 30 tokens are removed, and VDO filter is still applied in a more relaxed criteria than Liu et al. For diffs: They are split in blocks, location of change removed and kept filename, extension and context of change Subtokens split, non-English characters removed and all lowercased Diffs with more than 100 tokens are discarded and remaining tokenized using Word-PuntctTokenizer
Dataset Division	No mention of dataset division for this dataset in paper
Dataset available online	No link provided for the presented dataset

General Information			
Name of experiment	N/A	Authors	Shuyao Jiang
Year of publication	2019	Available online	Available at link https://github.com/ShuyaoJiang/CommitDataset
Publication paper name	Boosting Neural Commit Message Generation with Code Semantic Analysis	Notes on availability	Link contains files, link provided in paper

Experiment Information	
Comparison	
Treatments	Treatments used are 3 variations of Neural Machine Translation (NMT) proposed by Jiang et al.: Version 9.1 - Version with batch size 40 trained in proposed dataset for paper Version 9.2 - Version with batch size 15 trained in proposed dataset for paper Version 9.3 - Version with batch size 40 trained in raw version of commits instead of preprocessed
Ablation study	No ablation study performed in the paper
Human evaluation	No human evaluation performed in the paper
Metrics	BLEU-4, Modified N-gram precision for N=1,2,3,4
Other variables	No other statistical analysis performed
Training runs before reporting	No mention of training runs before reporting results
Comparison procedure	Paper takes NMT by Jiang et al. and applies ChangeScribe system to create the training dataset in order to improve metrics. The comparison procedure consists on score the 3 variations of NMT proposed in the presented dataset and present the results, followed by a brief discussion about the results
Experimental results	
Main results	Tailoring the inputs to the tasks lead to a increase in performance for the approaches using processed dataset compared to raw input
Other results	N/A
Experimental Setting	
Hardware	No mention of hardware used in training
Software	No mention of software used in training. Not possible to extract versions from code repository
Test Dataset	Test set for all the variations presented is the same test set resulting from dataset slit

Dataset	
Dataset Origin	Dataset is proposed by author of the paper in order to measure the results of the experiments on a larger corpus than Jiang et al. Furthermore, as dataset content needs to be preprocessed using ChangeScribe, raw datasets available are not useful
Dataset Composition	Dataset contains around 50,000 entries containing the preprocessed information for changes which was the output of ChangeScribe, and the corresponding human-written commit message from original repository, collected from 18 GitHub projects in Java with more than 20,000 stars **Variation of dataset containing raw commits used in the study**
Dataset Adjustment/Processing	Contents are extracted from commit histories of 18 selected Java projects Commit history is fed to ChangeScribe to generate the enumerations of code changes according to templates Generated outputs are filtered, removing labels and redundant statements, empty texts and texts with lenghts above 100
Dataset Division	Dataset is divided into: Training set - ~45,000 - (90%) Validation set - ~2,500 - (5%) Test set - ~2,500 - (5%)
Dataset available online	Dataset available at experiment link - https://github.com/ShuyaoJiang/CommitDataset

General Information			
Name of experiment	N/A	Authors	Xing Hu, QiuYuan Chen, Haoye Wang, Xin Xia, David Lo and Thomas Zimmermann
Year of publication	2022	Available online	Available at link https://github.com/xing-hu/DocEvaluation
Publication paper name	Correlating Automated and Human Evaluation of Code Documentation Generation Quality	Notes on availability	Files available, link provided in paper

Experiment Information	
Comparison	
Treatments	Treatments analyzed in the empirical study (for commit message generation) are: NMT version 10 - Not retrained, results reported evaluated over presented metrics NNGen version 10 - Input not fed, reported results evaluated over presented metrics PtrGNCMsg version 7 - Re-trained approach keeping original hyperparameters
Ablation study	No ablation study performed in the paper
Human evaluation	Human evaluation is the main purpose of the empirical study. To perform the study, 24 participants participated in a survey on two tasks: code comment generation and commit message geneation. Due to the scope of the study, only commit message generation is of interest. Each participants evaluate 50 results for all approaches over a corpus of 200 randomly selected entries. For each entry, participants evaluate in a scale 1 to 5 the following characteristics: naturalness, expresiveness, content adequacy, conciseness, usefulness and code understandability. After that, Kendall's Tau, Cohen's Kappa and Pearson Correlation Coefficient are computed for each of the automatic scores given and the human scores, both to correlate metrics with human judgement and to measure agreement between participants.
Metrics	BLEU-4, ROUGE-L, METEOR, CIDEr and SPICE
Other variables	Other statistical variables are Kendall's Tau, Cohen's Kappa and Pearson Correlation Coefficient on human evaluation results, average values, and inter-correlation between different metrics
Training runs before reporting	For NMT and NNGen, no single run is performed since only reported results are scored, for PtrGNCMsg, re-train process is performed, but no specification of runs before reporting results
Comparison procedure	Comparison procedure starts with evaluating selected approaches over the automatic metrics selected for the study. After it, human evaluation agreement is presented according to score distribution and Cohen's Kappa. Results on Kendall's Tau and PCC are presented as well as average scores for human evaluation, followed by the discussion about the correlation of automatic metrics and human judgement. Distributions are compared for both metrics and human evaluation, and map for Pearson Correlation Coefficient is performed correlating metrics and human results. Lastly, on already generated code documentation, automatic metric correlation is scored and results discussed
Experimental results	
Main results	From compared approaches in automatic metrics NNGen performs best in dataset (same used to outperform NMT in NNGen paper)
Other results	The more information carried from the input into the generated message, the more correlation, usefulness and helpful the generated documentation is SPICE metric has the lowest correlation, while CIDEr and BLEU correlation is almost perfect between them and good with human judgement
Experimental Setting	
Hardware	For PtrGNCMsg (only approach re-trained), NVIDIA Tesla T4 GPU with 16GB of VRAM used
Software	Linux server used. No further software specifications
Test Dataset	Test dataset also comes from presented dataset, same division as original paper. Test set used to score all results for commit message generation

Dataset	
Dataset Origin	Dataset used is a preprocessing adaptation from 2M Dataset proposed by Jiang and McMillan.
Dataset Composition	Content of dataset is 32,000 preprocessed pairs of <diff,message> extracted from top 1,000 java projects in order of stars, extracted from 2M+ commits from Jiang and McMillan dataset
Dataset Adjustment/Processing	Take datapoint from reference dataset of Jiang and McMillan Filter first sentence, remove commit ID, issue ID, merge and rollback commits Remove commits larger than 1Mb, filter out commits longer than 30 words (msg) or 100 tokens (diff) Apply a VDO (Verb-Direct Object) filter - using Standford CoreNLP
Dataset Division	Random division of dataset into: Training set - 26,000 pairs (81.25%) Validation set - 3,000 pairs (9.375%) Test set - 3,000 pairs (9.375%)
Dataset available online	Dataset available at link https://github.com/xing-hu/DocEvaluation

General Information			
Name of experiment	N/A	Authors	Samanta Dey, Venkatesh Vinayakarao, Monika Gupta and Sampath Dechu
Year of publication	2022	Available online	Available at https://github.com/CMGeval/Evaluating-CMG
Publication paper name	Evaluating Commit Message Generation: To BLEU Or Not To BLEU?	Notes on availability	Files accesible, link provided in paper

Experiment Information	
Comparison	
Treatments	Treatments analyzed on the paper are: NMT version 11 - Evaluated on a different metric so cannot know if already evaluated metrics correspond NNGen version 11 - Evaluated on a different metric so cannot know if already evaluated metrics correspond CommitGen version 3 - Evaluated on a different metric so cannot know if already evaluated metrics correspond
Ablation study	No ablation study performed in paper
Human evaluation	No human evaluation performed. Nevertheless, human evaluation in Tao et al. 2022 taken to consider human evaluation information
Metrics	Log-MNEXT (Variation of METEOR-NEXT metric)
Other variables	Spearman's Correlation between human evaluation and automatic metrics
Training runs before reporting	No mention of runs before reporting results for treatments in paper
Comparison procedure	Comparison procedure starts by correlating automatic metrics in their forumulation with presented results from Tao et al. study. After it, computation of added and deleted factors (such as length, word order...) is presented for various metrics formulas. Finally, as metrics are affected by correlation issues with human evaluation, table presented of the performance of approaches in new metric introduced (Log-MNEXT) is presented, and results discussed
Experimental results	
Main results	From evaluation in paper, NNGen outperform approaches in C++, JavaScript and Python, while CommitGen does in C# and NMT in Java
Other results	Factors - Length, Word Alignment, Semantic Scoring, Lower-Casing and Punctuation-Removal should be considered when designing metrics to evaluate commit message generation approaches
Experimental Setting	
Hardware	No mention of hardware used in paper
Software	No mention of software used in paper
Test Dataset	No mention of differences in test set from the one presented in the used dataset. Assumed test set comes from same dataset and division as presented

Dataset	
Dataset Origin	Large dataset called MCMD proposed by Tao et al. ICSME '21.
Dataset Composition	Contains filtered pairs <diff,message> from dataset proposed in mentioned paper. It contains entries in 5 programming languages (C++, C#, Java, Python and JavaScript) from top 100 projects on GitHub. For each language: C++ - ~200,000 entries C# - ~188,000 entries Java - ~200,000 entries Python - ~257,000 entries JavaSript - ~250,000 entries
Dataset Adjustment/Processing	Original dataset contains entries for each of the 5 programming languages Filter redundant messages such as merge and rollback commits Filter out noisy message as stated in Liu et al. 2018 To balance size data, they retain 450,000 commits for each programming language Authors filter out commits with multiple files and non-parseable files (.mp3,.jar...)
Dataset Division	Dataset is divided in the 5 programming languages (C++/C#/Java/Python/JavaScript): Training set - 160,948/149,907/ 160,018/206,777/ 197,529 - Total 875,179 (~80%) Validation set - 20,000/18,688/19,825/25,912/24,899 - Total 109,324 (~10%) Test set - 20,141/18,702/20,159/25,837/24,773 - Total 109,612 (~10%)
Dataset available online	Available at link https://github.com/CMGeval/Evaluating-CMG