NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

INTERDEPARTMENTAL MASTER'S PROGRAM

"LANGUAGE TECHNOLOGY"

THESIS

Constrained Text Generation

Nikolaos K. Katsifarakis

Supervisor: **Dimitris Galanis,** Researcher C' (ILSP)

ATHENS

OCTOBER 2022

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

"ΓΛΩΣΣΙΚΗ ΤΕΧΝΟΛΟΓΙΑ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παραγωγή Κειμένου Υπό Περιορισμούς

Νικόλαος Κ. Κατσιφαράκης

Επιβλέπων: **Δημήτρης Γαλάνης,** Ερευνητής Γ' (ΙΕΛ)

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2022

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Constrained Text Generation

Nikolaos K. Katsifarakis
A.M.: It1200009

SUPEVISOR: **Dimitris Galanis,** Researcher C' (ILSP)

EXAMINATION COMITEE:
Vassilis Papavassiliou, Research Associate (ILSP)
Haris Papageorgiou, Researcher A' (ILSP)

October 2022

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παραγωγή Κειμένου Υπό Περιορισμούς

Νικόλαος Κ. Κατσιφαράκης
Α.Μ.: It1200009

ΕΠΙΒΛΕΠΩΝ: **Δημήτρης Γαλάνης,** Ερευνητής Γ’ (ΙΕΛ)

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Βασίλης Παπαβασιλείου,** Συνεργαζόμενος Ερευνητής (ΙΕΛ)
Χάρης Παπαγεωργίου, Ερευνητής Α’ (ΙΕΛ)

Οκτώβριος 2022

ABSTRACT

Constrained text generation is a relatively new topic in NLG. It has been shown that pretrained Language Models (GPT-2, BART etc.), with no additional modifications or tuning, do not give good results. Many approaches have been introduced to improves results by utilizing external corpora, and by applying task – specific modifications to existing Language Models, as well as their fine – tuning process.

This study focuses on the task of producing phrases that contain a give set of words (concepts). For this purpose, we apply a heuristic scoring method, on top of the GPT-2 scoring function, in order to guide the production of the phrase towards said concepts, in a natural and semantically sound way.

More specifically, we add bonus scores to the top – scoring (according to GPT-2) candidate words, for example as a function of their PMI and embedding similarity to the remaining concept words (the containment of which, in the produced phrase, is the constraint), as well as a bonus if the candidate word itself is a concept. In order to test the models we developed, we employed the widely used, for this task, CommonGen dataset. The tests showed that the Beam Search algorithm, with a suitable objective function that combines the aforementioned heuristics, outperforms the Greedy search algorithm. Additionally, a variation of the Diverse Beam Search algorithms, that ensures diversity among the beams (possible solutions) further improves the results, when using a suitable objective function.

The employed heuristics were combined in two different ways: a) with a linear function, in which each heuristic is manually given and weight; b) with the application of Machine Learning methods, for automatically calculating the weights. More specifically, for method (b), the best results were achieved using k-NN regression and were comparable to the optimal results obtained using method (a), while method (b) has the advantage that it does not require testing different weights.

SUBJECT AREA: Natural Language Generation

KEYWORDS: NLG, constraints, concepts, decoding, objective function

ΠΕΡΙΛΗΨΗ

Η παραγωγή κειμένου υπό περιορισμούς αποτελεί έναν σχετικά νέο τομέα της Παραγωγής Φυσικής Γλώσσας. Προγενέστερα άρθρα έχουν δείξει πως τα προεκπαιδευμένα Γλωσσικά Μοντέλα (π.χ. GPT-2, BART) από μόνα τους και χωρίς τροποποιήσεις δεν προσφέρουν αρκούντως καλά αποτελέσματα. Πολλές προσεγγίσεις έχουν προταθεί για να πετύχουν καλύτερα αποτελέσματα, χρησιμοποιώντας εξωτερικές πηγές και πόρους, σε συνδυασμό με τροποποιήσεις στα υπάρχοντα Γλωσσικά Μοντέλα, καθώς και στην διαδικασία ρύθμισης τους για το συγκεκριμένο πρόβλημα.

Η παρούσα έρευνα επικεντρώνεται στο πρόβλημα της παραγωγής φράσεων που περιέχουν ένα δεδομένο σύνολο λέξεων (concepts). Για αυτόν τον σκοπό, εφαρμόζουμε ευριστικές μεθόδους βαθμολόγησης, επιπλέον της συνάρτησης βαθμολόγησης του γλωσσικού μοντέλου GPT-2, ώστε να καθοδηγήσουμε την παραγωγή της φράσης προς τις προαναφερθείσες λέξεις, με τρόπο φυσικό και νοηματικά ορθό.

Πιο συγκεκριμένα, προσθέτουμε επιπλέον βαθμούς στις υποψήφιες λέξεις με την καλύτερη GPT-2 βαθμολογία, π.χ. ως συνάρτηση του PMI τους, και της διανυσματικής τους ομοιότητας με τα εναπομείναντα concepts (για τα οποία υπάρχει ο περιορισμός να συμπεριληφθούν στη τελική πρόταση), καθώς και το κατά πόσο είναι οι ίδιες οι υποψήφιες λέξεις ένα από αυτά. Για όλες τις δοκιμές των συστημάτων που αναπτύχθηκαν, χρησιμοποιήσαμε το ευρέως χρησιμοποιούμενο για αυτό το πρόβλημα CommonGen dataset. Οι δοκιμές/πειράματα έδειξαν πως ο αλγόριθμος αναζήτησης Beam Search με μια κατάλληλη αντικειμενική συνάρτηση που συνδυάζει τις προαναφερθείσες ευριστικές δίνει βελτιωμένα αποτελέσματα σε σχέση με τον Greedy. Μάλιστα, μια συγκεκριμένη εκδοχή του ο Diverse Beam Search, που εξασφαλίζει την ποικιλομορφία μεταξύ των beams (πιθανές λύσεις), βελτιώνει περεταίρω τα αποτελέσματα με την κατάλληλη αντικειμενική συνάρτηση.

Οι ευριστικές που χρησιμοποιήθηκαν συνδυάστηκαν με δύο διαφορετικούς τρόπους, α)_με μια γραμμική συνάρτηση που σε κάθε ευριστική δίνεται ένα βάρος χειροκίνητα β) με την εφαρμογή μεθόδων Μηχανικής Μάθησης, ώστε να υπολογιστούν αυτόματα τα βάρη. Πιο συγκεκριμένα για το (β) τα καλύτερα αποτελέσματα επιτευχθήκαν με παλινδρόμηση με τη μέθοδο k-NN, τα οποία είναι συγκρίσιμα με τα βέλτιστα που πετύχαμε με την προσέγγιση (α), ενώ η προσέγγιση (β) έχει και ως πλεονέκτημα ότι δεν απαιτεί δοκιμή διαφορετικών βαρών.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Παραγωγή Κειμένου

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Παραγωγή Φυσικής Γλώσσας (NLG), περιορισμοί, αποκωδικοποίηση (decoding), αντικειμενική συνάρτηση (scoring/objective functions)

CONTENTS

PREFACE **11**

1. INTRODUCTION **12**

1.1 Aim of the study **12**

1.2 Related work **12**

1.2.1 Evaluation measures 12

1.2.2 CommonGen dataset 13

1.2.3 Generative Commonsense Reasoning methods 15

1.3 Contribution of the study **17**

2. PROPOSED GENERATION METHODS **18**

2.1 Background **18**

2.1.1 GPT 2 18

2.1.2 Greedy Decoding 18

2.1.3 Beam Search 19

2.1.4 GloVe 19

2.1.5 Word2Vec 19

2.1.6 PMI 20

2.2 Proposed generation methods **20**

2.2.1 Scoring/objective functions 20

2.2.2 Diverse Beam Search 22

3. EXPERIMENTS AND RESULTS ΣΦΑΛΜΑ! ΔΕΝ ΕΧΕΙ ΟΡΙΣΤΕΙ ΣΕΛΙΔΟΔΕΙΚΤΗΣ.

3.1 Preliminary experiments and results **23**

3.2 Experiments with Beam Search **24**

3.2.1 Experiments for finding optimal N 24

3.2.2 Experiments for finding optimal weights 25

3.2.3 Diverse Beam Search experiments 28

3.3 Overall results in dev set **30**

3.4 Learn weights with ML **31**

3.5 Evaluation on the test set of CommonGen **32**

3.5.1 Test set results evaluation 32
3.5.2 Error Analysis 33

4. CONCLUSIONS AND FUTURE WORK ..... ΣΦΑΛΜΑ! ΔΕΝ ΕΧΕΙ ΟΡΙΣΤΕΙ ΣΕΛΙΔΟΔΕΙΚΤΗΣ.

ACRONYMS.....35

REFERENCES 36

LIST OF FIGURES

Figure 1: Performance as a function of the number of top words selected 25

Figure 2: Influence of each parameter, with a subset of 100 concept sets 27

Figure 3: Performance with 10 groups of 3 beams, with different similarity bonus setups 29

LIST OF TABLES

Table 1: Statistics for the CommonGen dataset 13

Table 2: CommonGen Leaderboard: The models are sorted by SPICE score. 14

Table 3: Results on dev set of the comparison between GloVe and Word2Vec..... 23

Table 4: Performance as a function of the number of top words selected 24

Table 5: Performance of various setups, with a single group of 5 beams..... 26

Table 6: Performance of various setups, with a subset of 100 concept sets from dev part of CommonGen 26

Table 7: Performance with 10 groups of 3 beams, with different similarity bonus setups. The whole dev set was used 28

Table 8: Results of combinations of similarity bonuses on the whole dev. set 30

Table 9: Results of the application of our method, compared to vanilla GPT-2 30

Table 10: Results of the application of the k-NN calculated bonus..... 31

Table 11: Comparison of the test - set performance of our models and similarly structured methods. Table is sorted based on SPICE 32

PREFACE

The present study is a Master's thesis, for the "Language Technology" Master's program of the National and Kapodistrian University of Athens.

1. INTRODUCTION

1.1 Aim of the study

The aim of this thesis is to improve constrained text generation methods; specifically, the proposed method has been designed for the task of Generative Commonsense Reasoning (GCSR), i.e., the generation of coherent sentences or phrases with the constraint of including a set of given concept words. GCSR task was introduced by Lin et al. [1] along with a relevant benchmark dataset called “CommonGen”. For example, given the list of words: [‘kid’, ‘room’, ‘dance’], the desired output in CommonGen could be ‘A kid is dancing in the room.’ As mentioned by Lin et al., GCSR can be used for assessing commonsense reasoning ability, which so far is considered (to a large extent) unattainable for Artificial Intelligence (AI) and Natural Language Processing (NLP). Consequently, the resolution of this problem can contribute significantly to improving NLP applications, such as chatbots [2], question answering systems [3] etc.

1.2 Related work

1.2.1 Evaluation measures

The measures that are used for evaluating GCSR methods are the following:

- **BLEU**: Papineni et al. [4] proposed BiLingual Evaluation Understudy (BLEU) as an automated method of machine translation evaluation. It is calculated as the percentage of the n-grams of a reference sentence that are present within the produced sentence. Depending on the value of n, the BLEU variant is named accordingly, for example BLEU-3 calculates the percentage of 3- grams, etc.
- **ROUGE:** Lin [5] proposed Recall-Oriented Understudy for Gisting Evaluation (ROUGE) to automatically determine the quality of a machine – generated summary. It is based on n-gram similarity, and has several variants, such as ROUGE-N (N-gram Co-Occurrence), ROUGE-S (Skip-Bigram Co-Occurrence) etc. More commonly used for a task such as CommonGen is ROUGE-L, which measures the longest common subsequence between the produced sentence and the reference.
- **METEOR:** Metric for Evaluation of Translation with Explicit ORdering (METEOR) was proposed by Banerjee et al [6] as another metric for machine translation evaluation. It is based on unigram overlap; unigrams can be matched based on their surface forms, stemmed forms, and meanings. METEOR calculates a score based on unigram-precision, unigram-recall, and a measure designed to capture how well-ordered the matched words in the machine translation are in relation to the human reference.
- **CIDEr:** Consensus-based Image Description Evaluation (CIDEr) was proposed by Vedantam et al. [7] as an automatic evaluation measure for systems that generate image descriptions. Each automatically generated and reference sentence is represented by the set of n-grams that they contain. CIDEr takes into account n-gram frequency by using TF-IDF scores; i.e. n-grams that commonly appear in many images are given lower weight. CIDEr is calculated using cosine similarity on TF-IDF vectors constructed for the generated and reference sentences. Each vector position corresponds to a specific n-gram; therefore each vector represents all n-grams of length n for a sentence. For the final score a

combination of cosine similarity scores for n-grams vectors of varying lengths is used.

- **SPICE:** Originally proposed by Anderson et al. [8] as an automated caption evaluation metric, Semantic Propositional Image Caption Evaluation (SPICE) focuses on “semantic propositional content”. The reason for developing SPICE was to overcome the limitations of the other evaluation measures which are sensitive to n-gram overlap. Instead, SPICE is based on scenes graphs; a scene graph encodes the objects, attributes and relationships found in an image caption. A graph is constructed by exploiting the output of a dependency parser and SPICE is calculated as an F – score over the tuples of the candidate and reference semantic scene graph.
- **Concept Coverage:** the average percentage of input concepts that are present in lemmatizatized outputs.

As it is reported in Lin et al. [1] SPICE is the measure that correlates the most with human evaluations.

1.2.2 CommonGen dataset

CommonGen dataset [1] was created for training and evaluating GCSR systems. As already mentioned in GCSR given a set of common concepts (a.k.a. concepts set) the goal is to generate a coherent sentence describing an everyday scenario using these concepts.

The CommonGen dataset is split into train, dev, and test parts and in total it contains 35,141 concept sets; see Table 1 below.

Table 1: Statistics for the CommonGen dataset

Statistics	Train	Dev	Test
# Concept-Sets	32,651	993	1,497
Size=3	25,020	493	-
Size=4	4,240	250	747
Size=5	3,391	250	750

For each concept set of train and dev parts a set of human references are provided. E.g.

Concept set = {fall, ground, jump}

References = {“The girl may fall if she tries to jump to the ground.”, “A man jumping over a log falls to the ground.”, “Jump on the ground but don’t fall.”, “A man jumps a ramp and falls to the ground.”}

For the test part the references are not publicly available and for obtaining results the predictions have to be submitted (via email) to the research team that maintains the official Leaderboard [9]. In Table 1 a snapshot of the Leaderboard extracted in May 2022 from the respective web page is presented. Three models very similar to the ones that were developed in this thesis have been added in the table (rows 8, 13, 17) for direct comparison by consulting the relevant papers.

Table 2: CommonGen Leaderboard: The models are sorted by SPICE score.

Rank	Model	BLEU – 4	CIDEr	SPICE
-	Human	46.49	37.64	52.43
1	KFCNet	43.619	18.845	33.911
2	KGR[4]	42.818	18.423	33.564
3	KFC (v1)	42.453	18.376	33.277
4	R^3-BART	41.954	17.706	32.961
5	WittGEN + T5-large	38.233	18.036	31.682
6	I&V	40.565	17.716	31.291
7	RE – T5	40.863	17.663	31.079
8	**Neurologic - supervised**	**26.7**	**14.7**	**30.3**
9	A* Neurologic (T5-large)	39.597	17.285	30.130
10	VisCTG (BART-large)	36.939	17.199	29.973
11	SAPPHIRE (T5-large)	37.119	16.901	29.751
12	KG-BART	33.867	16.927	29.634
13	**A* Neurologic – unsupervised (greedy)**	**28.6**	**15.6**	**29.6**
14	EKI-BART	35.945	16.999	29.583
15	T5-Large	31.962	15.128	28.855
16	BART	31.827	13.976	27.995
17	**Neurologic - unsupervised**	**24.7**	**14.4**	**27.5**

18	UniLM	30.616	14.889	27.429
19	BERT-Gen	23.468	12.606	24.822
20	GPT-2	23.73	12.187	23.567
21	T5-Base	18.546	9.399	19.871

1.2.3 Generative Commonsense Reasoning methods

The GCSR task of is relatively new (Lin et al. [1]), however, a significant number of methods have already been proposed. Many of them use a retrieve-and-generation approach where prototype sentences (templates) are used, retrieved from external sources/corpora. For example, H. Wang et al. [10] proposed a T5 encoder-decoder architecture called retrieval-enhanced T5 (RE-T5) where retrieval methods were used for enhancing pre-training and fine-tuning steps of T5. Specifically, at pre-training step retrieval is used for creating auxiliary (prototype) sentences that along with the input concepts and target sentence are fed to T5. In a similar manner at fine-tuning stage retrieval is used for determining the top k sentence candidates for each concept set; then the candidates and input concepts are fed to the model. The application of RE-T5 on the CommonGen dataset showed that the results it produced (Table 2, row 7) were not only improved, compared to a vanilla T5 (Table 2, row 15 and 21), but are also comparable to those of other similar methods e.g. KG-BART and EKI-BART, rows 12 and 14 respectively. In a similar approach, Li et al. [11] proposed a method called "Knowledge Filtering and Contrastive Learning Network (KFCNet)" which uses a two-stage procedure for retrieving prototypes. Specifically, in stage 1, a sparse vector model is used for finding N candidates that contain the desired concepts from a corpus D. In stage 2, these candidates are scored using a trained multi-layer perceptron (MLP). Each candidate (S) is represented with the following sequence S = [CLS] + concept set + [SEP] + candidate + [SEP] and the BERT embedding vector of the [CLS] token is given as input to the MLP. Subsequently, the candidate with the highest score is fed to a BART model for generating the final output. For training BART contrastive learning is applied on both the encoding and decoding steps; e.g. on the former, the contrastive module helps to capture global target semantics. The BART model was coupled with a Beam Search algorithm (Beam size = 5) for finding non-greedy solutions. The results that they obtained (Table 2, row 3) with their model significantly outperformed the thus far state-of-the-art models (e.g. RE-T5). Liu et al. [12] propose another retrieve-and-generation approach, similar to KFCNet. Their Knowledge – enhanced Commonsense Generation framework consists of four stages: Retrieval, Retrospect, Refine and Rethink ("KGR^4"). First, it identifies relative sentences from external corpora, to use as prototypes based on H. Wang et al.'s [10] retrieval methods (RE-T5). The prototype sentences are ranked using a RoBERTa-based classifier and the top 3 are kept (as in Wang et al [10]). Then, these sentences are copied or edited, to create better generations (Retrospect); this is done by using a BART model. The Refine step is for fixing any potential errors in the sentences; again a BART model is used. Finally, the output sentence is selected (Rethink), from this set of candidate sentences based on the scores that are returned from the BART model. The authors' extensive experimentations with KGR^4 showed that it achieves high SPICE [8] values (Table 2, row 2), as well as competitive scores in the other commonly used metrics.

P. Wang et al [13] tackle the problem with a similar perspective (to the one described so far), however, they do not retrieve templates but exploit external knowledge for imagining the scene that is to be described by the produced sentence. Specifically, the "Imagine-and-Verbalize (I&V)" method that they proposed constructs ("imagines") a relational Scene Knowledge Graph (SKG), which identifies relations among the input concepts; i.e., the graph represents the background knowledge that is required for reasoning and generation. The module that constructs the SKG is trained on a set of SKG instances from different resources and modalities. The SKG is then leveraged as a constraint, during the generation (verbalization module) of a plausible scene description. The obtained results indicate that I&V is effective (see Table 2, row 6) for both concepts-to-sentence (i.e., GCSR) and concepts-to-story tasks [13].

Another family of Generative Commonsense Reasoning methods do not use external knowledge; e.g., retrieved prototype sentences or knowledge graphs. They start from scratch and are based solely on left-to-right decoding from language models (e.g. GPT-2). For example, Lu et al. [2] proposed an algorithm, called "NeuroLogic" Decoding. The aim of the method is to find a sequence that has the maximum possible fluency (based on GPT-2 scores), while at the same time satisfy the given constrains; the latter is achieved by adding a penalty score within the decoding objective function. A beam-search-based algorithm is used to select the optimal solution, and is designed to respect predicate logic constraints; i.e., boolean functions indicating the occurrence of phrase in a sequence. This approach led to a significant improvement in all commonly used performance metrics (ROUGE-L [5], BLEU [4] etc.), as well as the coverage of the constraints, compared to vanilla Beam Search decoding [2]. In their follow-up paper [14], Lin et al. aimed to improve NeuroLogic, by employing lookahead methods. For this purpose, they introduced "NeuroLogic*" a decoding algorithm, inspired by the A* algorithm [15]. This algorithm includes a future cost (in the objective function), that predicts constraint satisfaction thus guiding generation towards both coherence and completion of the GCSR task. The proposed methodology applied on top of Greedy decoding, further improved performance, compared to NeuroLogic Decoding on CommonGen corpus [14]. As shown in Table 2 the NeuroLogic and NeuroLogic* methods [2,14] achieve (see Table 2, rows 8, 13, 17) significantly lower scores than the retrieve-and-generation approaches that were described above. We conjecture that this is due to that the latter methods use external knowledge and/or start from a template while the former start from scratch and do not use additional data/resources. Even a variant[1] of NeuroLogic* that uses a T5-large model which is much bigger that GPT-2 (770 vs 1.5 million parameters) does not achieve to surpass retrieve-and-generation approaches (Table 2, row 9).

There are also papers that focus on specific enhancements/improvements of the generation algorithms. For example, Feng et al.'s [16] method (Table 2, row 11) focuses on finding a suitable order of the input concepts by using GPT-2 and perplexity measure, as well as augmenting the concept set with some more words (keywords), that would lead to a more natural sequence. Fan et al. [17] use external knowledge to implement two additional modules to the encoder – decoder models, namely scaling

[1] https://docs.google.com/document/d/1VaFJkXT0fLiJ40MPSvBmC1ZcNbbrG8J-TgvTQOyG15Y/edit

module and position indicator, for better identifying the relationships among the concepts, and thus achieve better semantic coherence (Table 2, row 14).

1.3 Contribution of the study

Although the number of studies that tackle constrained text generation is already significant, the fact remains that the task is still relatively new, and thus, there is still room for experimentation and improvement. In this study, we do not use retrieval methods for obtaining prototypes even though it has been proven that they give good results. This is due to the fact that these approaches a) depend on large external datasets that might not be always available (e.g., for a specific domain) and b) in several cases additional time has to be spent for training task or domain specific models.

Instead, we investigate a more generic approach, and we focus on the decoding part of the process; i.e., in designing a search algorithm that starts from scratch and employs a fine-tuned language model (LM) which along with an appropriate objective function steers generation towards desired outputs. More specifically, we use lookahead heuristics which in combination with a conditional LM (i.e., GPT-2), favor the words that either satisfy a constraint themselves, or lead to a word that does. Since our methodology follows the same approach as that of Lu et al., NeuroLogic and NeuroLogic* (which were described above) will be our main reference, regarding both architecture and results.

2. PROPOSED GENERATION METHODS

2.1 Background

In this chapter the machine learning models, word representation approaches, search-based algorithms etc. that were used for the development of the generation methods of this thesis are briefly presented.

2.1.1 GPT 2

Generative Pre-trained Transformer 2 (GPT-2) [18] was developed by OpenAI as a successor to the original GPT [19]. It is a transformer – based model [20] with 1.5 billion parameters. The "WebText" dataset that was used for training GPT-2 was compiled from reddit and it contains of 40 GB of text. The focus was on quality, e.g., only pages curated by humans were used.

GPT-2 has a decoder – only architecture and is trained for predicting the next word of a sequence, given all the previous ones. More specifically, a score is assigned to every candidate word, and subsequently, a probability is calculated, which could be formulated as:

$$p(output|input) \quad (1)$$

GPT-2 has been tuned (with appropriate data) and used in several downstream tasks, such as summarization, question answering etc. [18]. In these scenarios a more accurate formulation of the calculated probabilities would be:

$$p(output|input, task) \quad (2)$$

As it happens with other pre – trained models, GPT-2 performs better when dealing with common topics, while it struggles with domain – specific data. However, as already said it can be fine – tuned to not only better "understand" specific topics, but to also solve new tasks. This is crucial for our purpose, as it allows us to utilize the capabilities of such a powerful, general – purpose model for GCSR.

GPT-2 can be used along with a number of decoding/search methods, which can be selected though specific parameters, during the sequence generation process. The most widely known such methods are Greedy decoding, Beam Search, Top – K sampling and Top – P (nucleus) sampling [21]. The methods of this thesis were based on top of Greedy and Beam Search decoding and are described below.

2.1.2 Greedy Decoding

Greedy decoding, as well as any greedy algorithm, follows the strategy of making the locally optimal choice at each decoding step [22]. As such, it selects the token w_i with the highest conditional probability P(w_i | already generated sequence) in each step [23]. This is not optimal because for example, if words w_1, w_2 and $w_{3,,}w_n$ are the candidates in a specific step, and w_1 has the best conditional probability, it will be selected.

However, the selection of another token (e.g. w_2) might eventually lead to a better result. As such, greedy decoding is usually not the best option. However, it is relatively fast and effective enough to serve at least as a baseline.

2.1.3 Beam Search

In Beam Search (BS), (R. D. Raj [24]) as opposed to greedy decoding a number of best candidates, known as beam width (k), are selected and kept based on some score (e.g. GPT-2 conditional probability [25] at each step. The k best generated sequences continue to expand, until the “end” token (EOS) is reached. Beam Search (BS) usually outperforms Greedy decoding, if beam width, is appropriately chosen. However, BS becomes more computationally demanding, as the number of beams increases.

2.1.4 GloVe

Global Vectors for word representation (GloVe) is an unsupervised learning algorithm for obtaining vector representations for words, developed by Pennington et al. [26]. Given a corpus as a training dataset, GloVe learning procedure derives a model which assigns a N-dimensional vector to each word, based on its use in reference to the other words of the corpus. N is usually between 100 and 300. Such representations allow the calculation of linguistic or semantic similarity between words; e.g. by using cosine similarity between their corresponding vectors.

For training GloVe models, a matrix of word co – occurrence is required, of which only the non – zero entries are taken into account. The method requires the whole corpus to be scanned. This is a very demanding process; however, it only takes place once. Once the vectors are learnt, they can be saved and reused as needed, to avoid the aforementioned process.

A fact that signifies the consistency of a well – trained GloVe model, is that pairs of words that differ in the same semantic or linguistic way, have similar differences of their respective vectors. For example, the pairs man – woman and king – queen consist of words with that only differ in gender. A well – trained Glove model will produce vectors such that: $V_{man} - V_{woman} \approx V_{king} - V_{queen}$, where V_{man} is the vector corresponding to the word “man”, etc. In the same way, pairs of words, such as strong – stronger, quick – quicker will have similar differences between their respective vectors.

2.1.5 Word2Vec

Similar to GloVe is Word2Vec that was introduced by Mikolov et al. [27]. It is an algorithm that utilizes a Neural Network, for extracting word associations from large corpora. The words are represented in this model as vectors in an N – dimensional space as with GloVe. The number of dimensions, and the context window which determines how many words before and after are considered as context during the training procedure are parameters of the learnt model

After being trained, the word vectors can be used to measure the semantic similarity between words, i.e., by calculating the cosine similarity between their respective vectors as with GloVe.

2.1.6 PMI

Pointwise Mutual Information (PMI) is a measure that quantifies the likelihood of two words occurring in the same sequence [28]. More specifically, it is a measure of how often two words co – occur, in regards to how often each of these words co-occur by chance. Given the words w_1 and w_2, PMI is calculated with the following formula:

$$PMI(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_1)P(w_2)} \quad (3)$$

As was the case with GloVe, a corpus is required, from which the frequencies of the words, individually and together, can be extracted. Naturally, the larger the corpus the more accurately will the connection of each pair of words be quantified by PMI.

Regarding what qualifies as co – occurrence, a maximum accepted number of words between the examined pair is decided (window), as a parameter. This affects the resulting scores, as a number as small as 0 would only identify bigrams as co – occurrence, while a large number will identify cases where the two words happen to be in relatively close spots purely as a coincidence, without there being any semantic correlation that led the sequence from the first word to the second.

2.2 Proposed generation methods

2.2.1 Scoring/objective functions

Our methodology and scoring/objective is applied on top of the scoring process of the GPT-2 token generation step which is based on the conditional probability P(w_i | already generated sequence) [18]. Our objective aims to guide the search towards fulfilling the constraints (i.e., include the input concepts) while maintaining a semantically and linguistically correct output. That is achieved by giving extra credit in several ways to the words that are either one of the concepts themselves or is expected to lead to outputs that contain the concepts.

For our experiments we used the fine–tuned[2][3] to the CommonGen training set GPT-2 model of Lin et al. [2,14]. This fine – tuning process is very important, because the derived model is capable to guide towards the inclusion of the input concepts. In addition, it "familiarizes" the decoder with the way that this input is structured. The standard form of the input, as suggested by Lin et al. [1] is "$concept_1$, $concept_2$,…, $concept_n$ =", where n is the number of input concepts. Such an input, when given to a non fine–tuned GPT-2 model leads to unintelligible outputs, that include random characters and symbols, and are not resembling to natural language. On the other hand, the fine – tuned GPT-2 model without any further interventions, generates NL outputs after given the "=" symbol, and tends to include at least some of the input concepts.

As already mentioned, our methodology is directly applied to the word prediction/selection process. More specifically, we intervene right after all possible next

[2] https://github.com/GXimingLu/neurologic_decoding

[3] https://drive.google.com/drive/folders/1Jqav26p_g6BmpNg-6mx0AMiPYHH07Vju?usp=sharing

(candidate) words (w_i) have been assigned a conditional probability score $P(w_i \mid$ already generated sequence) by the GPT-2 decoder (GPT-2 score henceforth). In the greedy setting (see section 2.1.2) all possible candidates are equal to the size of vocabulary V (|V|) while in the beam search setting (see section 2.1.3) is k * |V|, where k is the size of the beam. Since it many cases it would be computationally expensive to calculate the additional bonus scores of every candidate word, we only calculate bonus scores to the N top ranking (according to GPT-2 score) candidates. From an engineering perspective N is an additional optional parameter in the implemented generation function. Optionally, before dealing with these top N candidates, we increase the scores of the remaining concept set (RC) directly.

The way that the scores of the top N word candidates is tampered, is that they get three different bonuses that operate as lookahead heuristics indicating if the selection of a w leads to the desired output. Specifically, the final score of every next top candidate word w of an already generated sequence (AGS) is calculated as follows:

$$Score_w = GPT2(w \mid AGS) + l_1 * PMI(w, RC) + l_2 * Emb.Similarity(w, RC) + l_3 * IsConcept(w, RC) \quad (4)$$

l_1, l_2 and l_3 are the weights of each bonus score and RC is the set of the remaining concepts; i.e., the ones that have not yet been included in the already generated output. The objective does not include a measure that "directly" measures the concepts that have been already included, however, a fine-tuned GPT-2 model is used which usually returns larger scores for the concept words.

The three lookahead heuristics scores are calculated as follows:

- The first is the similarity between the candidate word (w) and the remaining concept set (RC) is calculated using cosine measure over embeddings. Since, embeddings reveal of co – word occurrence (as also mentioned in Chapter 2.2), the idea is to increase the score of candidate words that are more likely to exist in the same contexts as the RC. The reason to select such words, is to guide the produced phrase towards the RC by creating a context in which they tend to occur. Also, the cosine similarity of a word with itself is equal to 1, which means that this bonus helps the concepts themselves to be selected.
- Similarly, the second bonus is the PMI score between the candidates and the remaining concepts (RC). PMI is even more specific about words coexisting in a sequence, within a pre–determined span of words. However, unlike embeddings-based similarity, and since a word is very unlikely to follow itself in a sequence, PMI does not directly encourage the selection of the remaining concepts themselves.
- Finally, the third bonus is a straightforward +1 to the score, multiplied by the corresponding factor, if the word is itself one of the remaining concepts. This is a very simple way to guide the decoder to prefer concepts over other words, however this might lead to bad outputs.

There are two different strategies, for calculating the first two lookahead scores (PMI and Emb.Similarity) for a word, given the RC set:

- Max: Calculate a score separately with each of the l remaining concepts and then select the maximum

- Ordered: Calculate a score with just one specific concept that represents RC. This strategy requires the concepts to be ranked by using some function.

2.2.2 Diverse Beam Search

The scoring methodology presented in the previous section can be used along with any search algorithm, e.g. Greedy and Beam Search (see sections 2.1.2 and 2.1.3). However, in the case of Beam Search the k best solutions (beams) very often are very similar to each other which in several cases leads to bad or not optimal results. In other words, the selection of a "good" word candidate at an early step might not lead eventually to good solutions.

To avoid such situations a variation of Beam Search has been used called Diverse Beam Search (DBS)[4] which has been proposed in [29]. The proposed DBS generates groups of beams that are diverse, by using a penalty score that discourages similarity between them. This way in our case (GCSR task), the generated groups lead to different phrases, thus increasing the likelihood that some of them will include all concept words in the right order as well as the required context words for creating a natural phrase. Our DBS returns the final result by ranking all output sequences from all groups by concept inclusion; if two sequences have the same number of concepts the one with the highest GPT-2 score is preferred. This way, we enforce the diversity of the beams, and pick the sequence that best fits the task at hand.

[4] Our implementation of DBS (as for BS) was based on code that is provided by HuggingFace: https://huggingface.co/docs/transformers/v4.21.1/en/main_classes/text_generation#transformers.generation_utils.GenerationMixin.

3. EXPERIMENTS AND RESULTS

As it is apparent from chapter 2, there is a great number of methods that had to be tested and several parameters of these methods need tuning, e.g., Greedy vs BS vs DBS, find optimal N for calculating lookahead heuristics, find optimal weights for lookahead heuristics (LH) scores, etc. For this purpose, a number of experiments was conducted, all of them in the dev set of CommonGen, where the reference sentences are available; see Section 1.2.2. In all experiments we have used the fine-tuned (on GCSR training data) GPT-2 model that was described in 2.1.1.

3.1 Preliminary experiments and results

While our focus is Beam Search (BS) since it is expected to give better results, we used Greedy decoding for some preliminary tests, because it is faster compared to BS, and therefore it is easier to get results and decide for some parameters.

First, we wanted to determine which embeddings similarity gives better results, GloVe or Word2Vec. For this we have set N = 10 and tested GloVe model (vector size = 300) vs Word2Vec model (vector size = 300) in 3 different cases; in each case different weights were used for the scoring function. Max strategy (see section 2.2.1) was used in all experiments for calculating embeddings similarity with the RC set. The results are presented in Table 3, where it is shown that GloVe consistently outperforms Word2Vec, and thus it was used in all subsequent experiments.

Table 3: Results on dev set of the comparison between GloVe and Word2Vec

Decoding method	ROUGE-L	BLEU 3	BLEU 4	Coverage
Greedy decoding / GPT-2	0.349	0.227	0.194	0.814
Greedy decoding / GPT-2 + 3 * GloVe + 3 * IsConcept	0.306	0.192	0.177	0.859
Greedy decoding / GPT-2 + 3* Word2Vec + 3 * IsConcept	0.303	0.183	0.169	0.835
Greedy decoding / GPT-2 + 2 * GloVe + 3 * IsConcept	0.313	0.186	0.168	0.969
Greedy decoding / GPT-2+ 2 * Word2Vec + 3 * IsConcept	0.309	0.179	0.16	0.941
Greedy decoding / GPT-2 + 3 * GloVe + 2 * IsConcept	0.321	0.211	0.197	0.958
Greedy decoding /	0.316	0.204	0.19	0.949

GPT-2+ 3 * Word2Vec + 2 * IsConcept				

The results show also that the usage of **IsConcept** significantly increases Coverage, especially when its weight is larger (as expected). However, the other measures (ROUGE, BLEU) drop when compared to Greedy decoding is used with GPT-2 scores.

3.2 Experiments with Beam Search

The experiments were performed using the Beam Search algorithm with five beams (k=5). For scoring all bonus factors (see section 2.2.1) were used; i.e., embedding-based cosine similarity, coverage bonus, and PMI scores. Additionally, the scores of the concept words were individually increased before the application of (4), as a way to ensure that they will be in the top candidates, thus guiding the produced phrase to include them.

3.2.1 Experiments for finding optimal N

A number of experiments were also performed for determining N, the number of top – scoring words that would be subject to LH bonuses scores (see Section 1). For these experiments we used $l_1 = 0$, $l_2 = 2$ and $l_3 = 3$ the best configuration in terms of Coverage of the previous section. This has been done to assess if BS can improve ROUGE and BLEU scores while keeping Coverage high. Larger values of N would likely benefit the quality of the results, however they would also significantly increase the required execution time. Therefore, we looked for the threshold, after which any improvement with the results would not be significant. The results are presented in Table 4 and Figure 1.

Table 4: Performance as a function of the number of top words selected

	Number of top words selected	**ROUGE-L**	**BLEU-3**	**BLEU-4**	**Coverage**
Greedy	**N/A**	0.349	0.227	0.194	0.814
Greedy	10	0.313	0.186	0.168	0.969
Beam Search	10	0.3692	0.2559	0.2201	0.832
Beam Search	30	0.3699	0.2616	0.2235	0.944
Beam Search	50	0.3701	0.2617	0.2237	0.946
Beam Search	70	0.3702	0.2617	0.2237	0.947

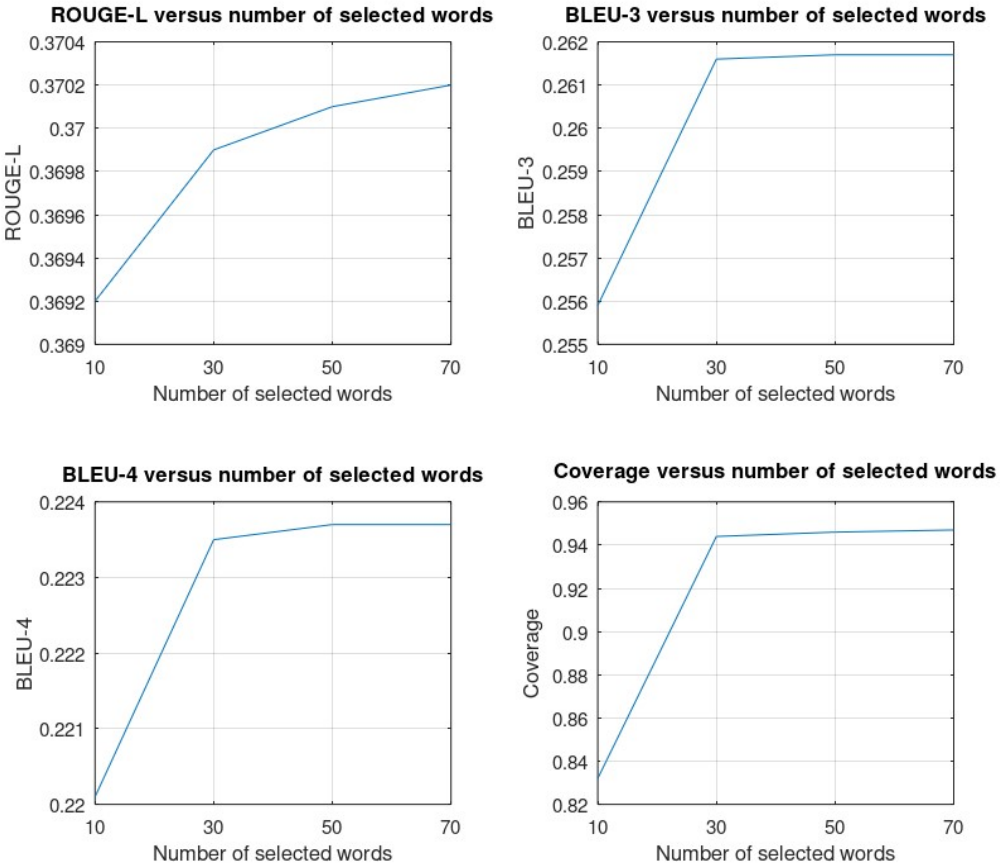

Figure 1: Performance as a function of the number of top words selected

While the increase in the performance, especially regarding the coverage of the constraints, is significant between 10 and 30 words, after 30 it is almost stable. Therefore, for all experiments of the following sections, the number of selected words was 30. Another interesting conclusion is that all configurations of BS (N=10, …, 70) achieve better results in terms of ROUGE and BLEU than Greedy Decoding with the same objective (and N = 10). At the same time Coverage remains remarkably high. This shows that BS is more robust and is capable to avoid aggressively selecting concept words and thus generate outputs that have high Coverage but low ROUGE and BLEU.

3.2.2 Experiments for finding optimal weights

The next step was to estimate the ideal configuration, in other words, the parameters by which each bonus would be multiplied, as well as the added score for the concepts. Our PMI model was trained using the Reuters corpus of the Natural Language ToolKit (NLTK) [30], with a window of 5 words. The results are presented in Table 5, where l_1, l_2 and l_3 are the parameters of Equation 4.

Table 5: Performance of various setups, with a single group of 5 beams

l_1	l_2	l_3	concept score increase	ROUGE-L	BLEU-3	BLEU-4	Coverage
0	0	0	0	0.370637	0.25644	0.22070	0.82206
0.5	4	4	0	0.328366	0.21470	0.19011	0.985803
0.6	4	4	0	0.328324	0.21593	0.19121	0.985853
0.7	4	3	0	0.334797	0.22082	0.19417	0.985501
1	4	2	4	0.285261	0.18169	0.15814	0.969943
2	4	3	4	0.330003	0.21694	0.18874	0.981687
0	2	0	3	0.370227	0.25151	0.21897	0.846472
0	2	0	0	0.375481	0.26092	0.22505	0.869607
0	1.5	0	0	0.372609	0.25860	0.22325	0.848925
0	1	0	4	0.365344	0.24797	0.21637	0.811979
0.1	2	0	3	0.368677	0.25096	0.21823	0.842893

While concept coverage is a very important part of the task, when it is “forced” via isConcept, it leads to unnatural sentences, as is shown in the results. This is caused mainly by the coverage bonus (IsConcept), and secondarily by the direct boost to concepts, which however does not increase the coverage.

To confirm this trend, and gain some insight of the ideal setup, more extensive experiments were performed on a subset of 100 randomly selected concept sets of CommonGen dev set, for assessing the impact of each parameter. The results are presented in Table 6 and Figure 2.

Table 6: Performance of various setups, with a subset of 100 concept sets from dev part of CommonGen

l1	l2	l3	Concept score increase	ROUGE-L	BLEU-3	BLEU-4	Coverage
0	0	0	0	0.389935	0.263466	0.228799	0.811594
0.5	0	0	0	0.353115	0.263118	0.223513	0.768116
1	0	0	0	0.280957	0.193863	0.170756	0.666667
1.5	0	0	0	0.170143	0.105709	0.093618	0.463768

2	0	0	0	0.152626	0.101534	0.083523	0.395623
0	1	0	0	0.388814	0.26335	0.227717	0.855072
0	2	0	0	0.396007	0.286642	0.247338	0.855072
0	3	0	0	0.383659	0.282942	0.243429	0.884058
0	4	0	0	0.382075	0.278832	0.243294	0.869565
0	0	1	0	0.35718	0.249197	0.216689	0.947566
0	0	2	0	0.351757	0.246141	0.216084	0.973783
0	0	3	0	0.338662	0.23495	0.20923	0.985019
0	0	4	0	0.334985	0.226998	0.20209	0.985019
0	0	0	1	0.369438	0.234872	0.201165	0.797101
0	0	0	2	0.356245	0.244068	0.211985	0.826087
0	0	0	3	0.369242	0.267662	0.234025	0.913043
0	0	0	4	0.35397	0.261061	0.224871	0.811594

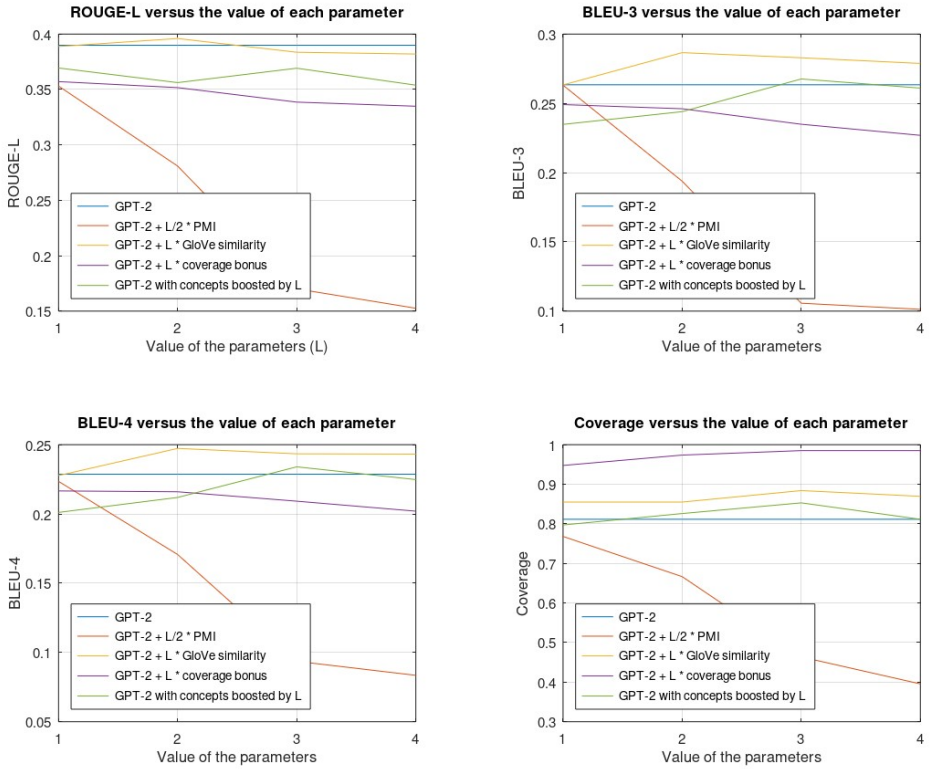

Figure 2: Influence of each parameter, with a subset of 100 concept sets

As can be seen, the usage of PMI decreases the score of all metrics, likely to inadequate training. Increasing the weight of IsConcept slightly improves concept coverage, however it causes a decrease in the other metrics. Concept bonus behaves in a similar way, greatly increasing concept coverage, but decreasing the other metrics.

On the other hand, GloVe similarity offers similar ROUGE-L with simple GPT-2 generation, and an increase in all other metrics. Specifically, when it is respective weight is equal to 2, all metrics are improved. Thus, this is the setup that was chosen for the experiments that followed.

3.2.3 Diverse Beam Search experiments

When BS uses just one group of beams it generates outputs that are very similar, so even having the model return them, and then pick based on concept coverage, had little to no impact to the results. Therefore, in order to improve the performance, we decided to experiment with more groups of beams and introduce diversity (see 2.4). Specifically, we chose a setup of DBS with 10 groups of 3 beams each, with a “diversity penalty” equal to 3. The only lookahead applied was 2*GloVe similarity which was found to give good results in the case of BS; see previous section 3.2.2.

An alternative was also tested. While thus far we used max strategy for embedding similarity, we now tried to order the concepts first, and only apply bonus for similarity with the concept that was next in the specified order (Ordered strategy). The order was determined by maximizing the sum of similarities between all adjacent concepts, once using their lemma, and once using all possible inflections. Both orders were obtained with exhaustive brute force search. The results of the subsequent experiments are presented in Table 7 and Figure 3, with vanilla DBS as reference (row 1).

Table 7: Performance with 10 groups of 3 beams, with different similarity bonus setups. The whole dev set was used

Similarity	l2	ROUGE-L	BLEU-3	BLEU-4	Coverage
Not used	0	0.375404	0.264742	0.226568	0.961341
Max strategy	2	0.388133	0.273323	0.235714	0.981132
strategy Ranked remaining concepts (lemma) by maximizing the sum of GloVe similarities	2	0.378056	0.264444	0.227301	0.985064
strategy - Ranked remaining concepts (all inflections) by maximizing the sum of GloVe similarities	2	0.370314	0.258456	0.221221	0.960349

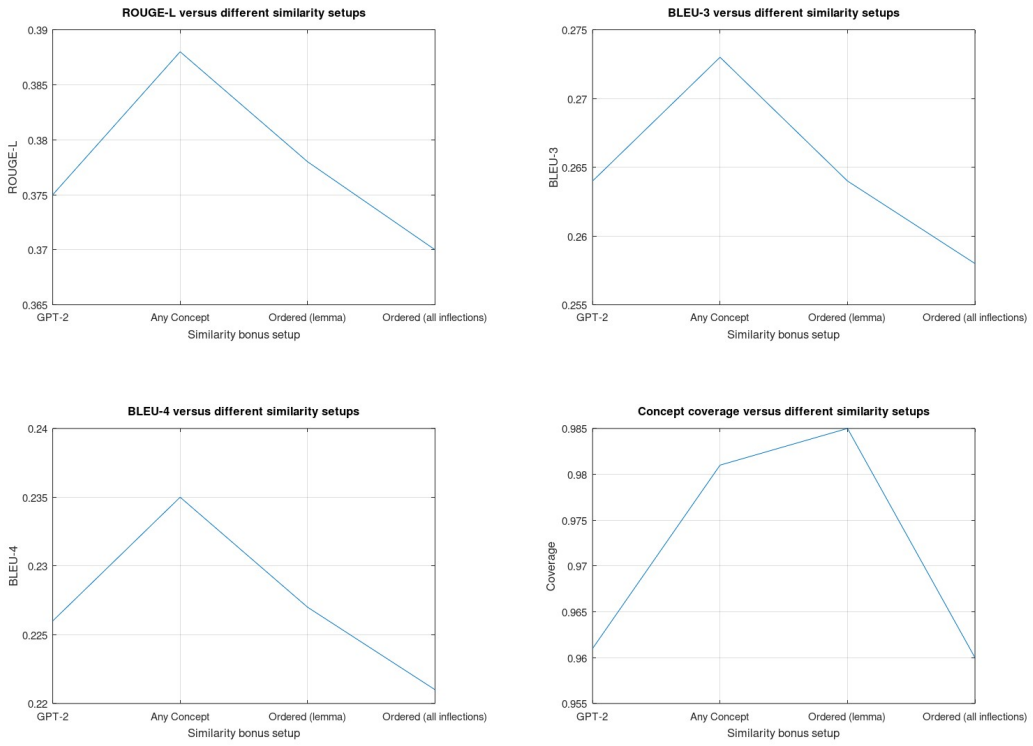

Figure 3: Performance with 10 groups of 3 beams, with different similarity bonus setups

As presented in the results, the overall optimal setup is the one where similarity is calculated as the maximum with any of the remaining concepts (Max strategy). While ordering the concepts was promising, it did not provide better scores.

In order to understand the reason, we generated different orderings for all concept sets of the dev. part of CommonGen and compared to that of the reference sentences. In the original order, (the one given as input) about 18.6% of the cases had at least one of the reference sentences that followed it. This percentage was raised to 25.4% with the ordering by lemma form similarity, and to 24.5% by all inflection's similarity. This showed that similarity leads to more realistic word order, than the random original one. However, it may be the case that even 25.4% is not enough, in the sense that forcing this order will lead to a sentence similar to one of the references only about once every 4 cases.

Inspired by Zhao et al [31], we used our best – performing model (DBS, 10 groups of 3 beams, l_2 = 2) to obtain orderings. This ordering had at least one reference sentence following it at 45.7% of the cases; significantly better than our previous methods. Therefore, new experiments were performed to assess whether it helps during decoding.

To avoid forming sentences that blindly focus at one concept at a time, making it difficult to move to the next concept, we tried a combination of the maximum similarity with any concept bonus, and a similarity bonus to only the concept that is next in the acquired order. The results are presented in Table 8.

Table 8: Results of combinations of similarity bonuses on the whole dev. set

Similarity Combination	ROUGE-L	BLEU-3	BLEU-4	Coverage
2*Max strategy + 0*Ordered	0.38838455	0.270569	0.233292	0.983132
0*Max strategy + 2* Ordered	0.38463645	0.266908	0.22958	0.983333
1*Max strategy + 1*Ordered	0.38614043	0.271743	0.235298	0.964718
1.33*Max strategy + 0.66*Ordered	0.39039677	0.274036	0.237374	0.97006
0.66*Max strategy + 1.33*Ordered	0.38725821	0.273579	0.235371	0.974345

As can be seen in Table 8, the combinations produced largely similar results. However, we managed to slightly improve on our previous results, in terms of ROUGE-L and BLEU scores.

3.3 Overall results in dev set

To summarize the results of our methods, in Table 9 we present the scores of our three best configurations, compared to two competitive baselines that use just GPT-2 for scoring (rows 1 and 2).

Table 9: Results of the application of our method, compared to vanilla GPT-2

GPT -2 Configuration	Objective	ROUGE-L	BLEU-3	BLEU-4	Coverage
Greedy decoding	GPT-2	0.349	0.227	0.194	0.814
Beam Search (1 group, 10 beams)	GPT-2	0.37063	0.2564	0.2207	0.82206
Beam Search (1 group, 10 beams)	GPT-2 + 2*GloVe (Max strategy)	0.37548	0.2609	0.225	0.8696
DBS (10 groups, 3 beams each)	GPT-2	0.37540	0.2647	0.2266	0.961341
DBS (10 groups, 3 beams each)	GPT-2 + 2*GloVe (Max strategy)	0.38838	0.2706	0.2333	0.983132
DBS (10 groups, 3 beams each)	GPT-2 + GloVe (1.33*Max strategy + 0.66*Ordered)	0.39039	0.274	0.2374	0.97006

Beam search outperforms Greedy decoding (as expected), and DBS further improves the results, especially in terms of concept coverage. The application of the GloVe similarity bonus provides a significant increase in all metrics, with the configuration that takes concept order into consideration scoring slightly higher in terms of ROUGE and BLEU.

3.4 Learn weights with ML

For automating the calculation of the total bonus score and avoid testing a large number of configurations (weight combinations) for the bonus scores, we employed Machine Learning. For this reason, we created a dataset of 5285 instances of randomly selected phrases being produced by our optimal model (last row of Table 9); these phrases correspond to a generation path.

For each phrase we calculated the following features: total number of concepts included, number of remaining concepts, PMI, GloVe similarity and IsConcept both max strategy and ordered as well as the GPT-2 score. The target prediction score was $(ROUGE-L+BLEU-3+BLEU-4)_{phrase+w}$ - $(ROUGE-L+BLEU-3+BLEU-4)_{phrase}$, in order to assess whether w increases (or decreases) the similarity between the phrase and the reference sentences.

Initially we tried a Neural Network [32] and a Decision Tree [33], but they didn't achieve good results, presumably due to the low amount of data. In addition, the Pearson's correlation between the predicted score and ideal score was rather low (~0.3) for the method. We achieved significantly better results with k - Nearest Neigbours (k-NN) regression [34], with k = 1000 which seems to require less data, Pearson's correlation was also improved (~0.58).

The comparison of our method that uses DBS along with k-NN for bonus prediction vs. simple Greedy decoding and the configurations that linearly combine bonus scores is presented in Table 10.

Table 10: Results of the application of the k-NN calculated bonus

GPT -2 Configuration	Objective	ROUGE-L	BLEU-3	BLEU-4	Coverage
Greedy decoding	GPT-2	0.349	0.227	0.194	0.814
Beam Search (1 group, 10 beams)	GPT-2	0.37063	0.2564	0.2207	0.82206
Beam Search (1 group, 10 beams)	GPT-2 + 2*GloVe (Max strategy)	0.37548	0.2609	0.225	0.8696
DBS (10 groups, 3 beams each)	GPT-2	0.37540	0.2647	0.2266	0.961341
DBS (10 groups, 3 beams each)	GPT-2 + 2*GloVe (Max strategy)	0.38838	0.2706	0.2333	0.983132

DBS (10 groups, 3 beams each)	GPT-2 + GloVe (1.33*Max strategy + 0.66*Ordered)	0.39039	0.274	0.2374	0.97006
DBS (10 groups, 3 beams each)	GPT-2 + k-NN-calculated bonus	0.38478	0.2723	0.2346	0.964651

While improving, compared to simple GPT-2, the k-NN based method is not as good as our optimal one.

3.5 Evaluation on the test set of CommonGen

3.5.1 Test set results evaluation

The final step was to evaluate the performance of our models on the test set of the CommonGen dataset. We submitted two versions of our DBS method (10 groups, 3 beams each): The first is the optimal one (manually assigned weights, GPT-2 + GloVe [1.33*Max strategy + 0.66*Ordered]), which we named “Linear DBS” since it combines heuristics with a linear function. The second is the k-NN based method (GPT-2 + k-NN-calculated bonus), which we named “k-NN DBS”.

Below in Table 11 we compare the results of our models with a GPT-2 approach and T-5 baselines, as well as all the best variations of Neurologic and Neurologic*, as they follow a similar approach to ours.

Table 11: Comparison of the test - set performance of our models and similarly structured methods. Table is sorted based on SPICE

Model	**BLEU – 4**	**CIDEr**	**SPICE**	**Coverage**
Neurologic - supervised	26.7	14.7	30.3	97.7
A* Neurologic (T5-large)	39.597	17.285	30.130	N/A
A* Neurologic – unsupervised (greedy)	28.6	15.6	29.6	97.1
Neurologic - unsupervised	24.7	14.4	27.5	96.7
Linear DBS	**21.004**	**12.041**	**24.406**	**96.03**
k-NN DBS	**20.844**	**11.6343**	**23.955**	**94.56**
GPT-2 (Beam Search, 5 beams as reported in [1])	23.73	12.187	23.567	79.09
GPT-2 (Beam Search, 5 beams as submitted by us)	**18.468**	**9.9238**	**21.97**	**85.02**
T5-Base (Beam Search, 5 beams as reported in [1])	18.546	9.399	19.871	76.67

Both versions of our method (linear and k-NN) surpass in SPICE two strong baselines that use Beam Search (5 beams); the one uses a T5-Base model [1] and the other uses a GPT-2 model [1]. Two versions of the latter baseline are included in the results; the first is described in [1] and presented in the CommonGen leaderboard and the second was implemented by us (our setup/configuration). Also, our methods (linear and k-NN) have significantly better Coverage than the aforementioned baselines. Interestingly, in terms of BLEU they are outperformed by the CommonGen GPT-2. Also, all variations of Neurologic score significantly higher in all metrics except Coverage where the results are comparable.

3.5.2 Error Analysis

Using a subset of 100 randomly selected phrases that the best configuration of our method (Linear DBS) produced for the test set, we looked for common errors; two types of such errors were identified. The first is when concepts are introduced to the phrase in a way that lacks semantic coherence, for example:

- [ride, board, water, boat] => boat and person board a water taxi for a ride on the river
- [shave, stand, leg, bathtub] => an old man with shaved legs standing in a tub with bathtubs

This type of error is much more common and pronounced when IsConcept bonus is applied, as it forces the selection towards concepts.

The second type of error is when concepts appear more than once, making the phrase unnatural. For example:

- [lawn, mower, mow, push] => a man pushing a mower in a lawn and mowing the lawn
- [sit, table, light, candle] => man sitting at table with candles and lighted candles

However, this type of error cannot be attributed solely to our method, as it only applies bonus to remaining concepts. Rather, it probably occurs to some extent due to the way that GPT-2 has been fine – tuned; i.e. to favor concepts, when producing the phrase.

Many of the examined phrases, though, are semantically and grammatically sound. For example:

- [cream, shave, face, apply] => a man applies cream to his face before shaving it
- [lunch, eat, worker, sit] => young woman sitting and eating lunch in a restaurant with other workers

Such phrases fulfil all the requirements of the task, as they contain all the concepts in a natural way.

4. CONCLUSIONS AND FUTURE WORK

This thesis developed methods for the task of Generative Commonsense Reasoning (GCSR). We have experimented with various search algorithms and scoring functions and it has been shown that

- The use of Diverse Beam Search (DBS) improves results over a vanilla Beam Search. This validates the findings of Neurologic papers, i.e., diverse partial solutions should be generated in the search space. To the best of our knowledge, we are the first to use DBS for GCSR.
- The use of embeddings similarity in the scoring function helps in generating better generation paths, i.e., it favors, in a natural way, the inclusion of words that lead to concepts.
- The use of a plausible ordering for the concepts helps in improving results. This also validates findings of previous papers.
- The two methods that were produced (Linear DBS, K-NN DBS) improve in terms of SPICE and Coverage two very strong baselines that use large LMs; especially in Coverage the difference is over 10-15% (depending on which setup we use). SPICE is the measure that correlates better with human judgements.

Potential future work would entail applying our method on other text generation tasks, for example automatic recipe generation [35], for testing its generalization abilities. It would also be sensible to apply our method with other LMs (BART [36], T5 [37] etc.), which could lead to better performance than GPT-2.

Finally, both aforementioned cases, as well as further optimizing our method with GPT-2 would benefit from improving the automated bonus score calculation process. The application of k – NN showed promising results, and a different, or better trained ML method could be on par, or even offer better results, than that of our hand – picked values for l_1, l_2 and l_3. Also, k-NN or any other ML method could be trained with more data.

ACRONYMS

NLP	Natural Language Processing
KFCnet	Knowledge Filtering and Contrastive learning Network
KGR^4	Knowledge – enhanced Commonsense Generation
I&V	Imagine-and-Verbalize
SKG	Scene Knowledge Graph
T5	Text-to-Text Transfer Transformer
RE – T5	Retrieval – Enhanced T5
BART	Bidirectional and Auto-Regressive Transformers
SAPPHIRE	Set Augmentation and Post-hoc PHrase Infilling and Recombination
BLEU	BiLingual Evaluation Understudy
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
METEOR	Metric for Evaluation of Translation with Explicit ORdering
CIDEr	Consensus-based Image Description Evaluation
SPICE	Semantic Propositional Image Caption Evaluation
GPT	Generative Pre-trained Transformer
BPE	Byte Pair Encoding
GloVe	Global Vectors (for Word Representation)
PMI	Pointwise Mutual Information
k-NN	k – Nearest Neighbors

REFERENCES

[1] B.Y. Lin, W. Zhou, M. Shen, P. Zhou, C. Bhagavatula, Y. Choi, X. Ren, CommonGen: "A constrained text generation challenge for generative commonsense reasoning", *arXiv preprint arXiv:1911.03705*, Nov. 2019

[2] X. Lu, P. West, R. Zellers, R.L. Bras, C. Bhagavatula, Y. Choi, "Neurologic decoding:(un) supervised neural text generation with predicate logic constraints", *arXiv preprint arXiv:2010.12884*, Oct. 2020

[3] A. Lazaridou, E. Gribovskaya, W. Stokowiec, N. Grigorev, "Internetaugmented language models through few-shot prompting for open-domain question answering", *arXiv preprint arXiv:2203.05115,* 2022.

[4] K. Papineni, S. Roukos, T. Ward, W.J. Zhu, "Bleu: a method for automatic evaluation of machine translation", *InProceedings of the 40th annual meeting of the Association for Computational Linguistics*, Jul 2002 (pp. 311-318).

[5] C.Y. Lin, "Rouge: A package for automatic evaluation of summaries", *InText summarization branches out*, Jul. 2004 (pp. 74-81)

[6] S. Banerjee, A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments", *InProceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, Jun. 2005 (pp. 65-72)

[7] R. Vedantam, C. Lawrence Zitnick, D. Parikh, "Cider: Consensus-based image description evaluation", *InProceedings of the IEEE conference on computer vision and pattern recognition,* 2015 (pp. 4566-4575)

[8] P. Anderson, B. Fernando, M. Johnson, S. Gould, "Spice: Semantic propositional image caption evaluation", *InEuropean conference on computer vision*, Oct 2016 (pp. 382-398)

[9] https://inklab.usc.edu/CommonGen/leaderboard.html (accessed Oct 10, 2022)

[10] H. Wang, Y. Liu, C. Zhu, L. Shou, M. Gong, Y. Xu, M. Zeng, "Retrieval enhanced model for commonsense generation", *arXiv preprint arXiv:2105.11174*, May 2021

[11] H. Li, Y. Gong, J. Jiao, R. Zhang, T. Baldwin, N. Duan, "Kfcnet: Knowledge filtering and contrastive learning network for generative commonsense reasoning", *arXiv preprint arXiv:2109.06704*, Sep. 2021

[12] X. Liu, D. Liu, B. Yang, H. Zhang, J. Ding, W. Yao, W. Luo, H. Zhang, J. Su, "KGR^ 4: Retrieval, Retrospect, Refine and Rethink for Commonsense Generation", *arXiv preprint arXiv:2112.08266*, Dec. 2021

[13] P. Wang, J. Zamora, J. Liu, F. Ilievski, M. Chen, X. Ren, "Contextualized Scene Imagination for Generative Commonsense Reasoning", *arXiv preprint arXiv:2112.06318*, Dec. 2021

[14] X. Lu, S. Welleck, P. West, L. Jiang, J. Kasai, D. Khashabi, R.L. Bras, L. Qin, Y. Yu, R. Zellers, N.A. Smith, "NeuroLogic A*esque Decoding: Constrained Text Generation with Lookahead Heuristics", *arXiv preprint arXiv:2112.08726*, Dec. 2021

[15] P.E. Hart, N.J. Nilsson, B. Raphael, "A formal basis for the heuristic determination of minimum cost paths" *IEEE transactions on Systems Science and Cybernetics*, 4(2):100-7, Jul. 1968

[16] S.Y. Feng, J. Huynh, C. Narisetty, E. Hovy, V. Gangal, "SAPPHIRE: Approaches for Enhanced Concept-to-Text Generation", *arXiv preprint arXiv:2108.06643*, Aug. 2021

[17] Z. Fan, Y. Gong, Z. Wei, S. Wang, Y. Huang, J. Jiao, X. Huang, N. Duan, R. Zhang, "An enhanced knowledge injection model for commonsense generation", *arXiv preprint arXiv:2012.00366*, Dec. 2020

[18] https://openai.com/blog/better-language-models/ (accessed June 7, 2022)

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, "Language models are unsupervised multitask learners", *OpenAI blog*,1(8):9, Feb. 2019

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, "Attention is all you need", *Advances in neural information processing systems 30*, 2017

[21] https://huggingface.co/blog/how-to-generate (accessed August 19, 2022)

[22] https://xlinux.nist.gov/dads//HTML/greedyalgo.html (accessed June 7, 2022)

[23] R. Sennrich, B. Haddow, A. Birch, "Neural machine translation of rare words with subword units", *arXiv preprint arXiv:1508.07909*, Aug. 2015

[24] R. D. Raj, "Speech understanding systems: A summary of results of the five-year research effort. department of computer science.", 1977

[25] https://www.width.ai/post/what-is-beam-search (accessed June 7, 2022)

[26] J. Pennington, R. Socher, C.D. Manning, "Glove: Global vectors for word representation" *In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP*) (pp. 1532-1543), Oct 2014

[27] T. Mikolov, K. Chen, G. Corrado, J. Dean, "Efficient estimation of word representations in vector space", *arXiv preprint arXiv:1301.3781* Jan 2013

[28] K. Church, P. Hanks, “Word association norms, mutual information, and lexicography” *Computational linguistics,16(1):22-9*, 1990
[29] A. K. Vijayakumar, M. Cogswell, R. R. Selvaraju, Q. Sun, S. Lee, D. Crandall, D. Batra, “Diverse beam search: Decoding diverse solutions from neural sequence models”, *arXiv preprint arXiv:1610.02424*, Oct 2016
[30] https://www.nltk.org/book/ch02.html (accessed August 19, 2022)
[31] C. Zhao, F. Brahman, T. Huang, S. Chaturvedi, “Revisiting Generative Commonsense Reasoning: A Pre-Ordering Approach”, *arXiv preprint arXiv:2205.13183,* May 2022
[32] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html (accessed October 1, 2022)
[33] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html (accessed October 1, 2022)
[34] N.S. Altman, “An introduction to kernel and nearest-neighbor nonparametric regression”, *The American Statistician: 175-85,* Aug 1992
[35] H. H. Lee, K. Shu, P. Achananuparp, P. K. Prasetyo, Y. Liu, E. P. Lim, L. R. Varshney, “RecipeGPT: Generative pre-training based cooking recipe generation and evaluation system”, *InCompanion Proceedings of the Web Conference 2020 pp. 181-184*, Apr 2020
[36] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension”, *arXiv preprint arXiv:1910.13461*, Oct. 2019
[37] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P.J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer”, *arXiv preprint arXiv:1910.10683*, Oct. 2019