3.4 Paragraph Generation

3.4.1 Templates

We use sentence and paragraph templates to make text grammatically correct. We create 90 different combinations of relations in ConceptNet in appendix F. The combination of two relations which called compound relation. For example, the combination of AtLocation-HasProperty is "[公園] 的 [貓] 是 [黑色] 的". However, only few of them can be used in paragraph templates. The number of combinations in relation "MadeOf", "PartOf", "SymbolOf", "MayUse" and "HasA" are 39. But the common sense of these relations are difficult to be applied to paragraphs. Some of these templates are not possibly to be said in daily life or there are no appropriate relations can match with. For example, the compound relation of HasSubevent-MayUse is "[約會][喝咖啡] 時會用到 [杯子]". This sentence is semantically and grammatically correct, but it is less likely in our daily life conversations. Therefore, we only use 23 different compound relations in paragraph templates.

We create 8 different paragraph templates in appendix F. They are combinations of sentence templates. One of the example in table 3.3. The number of

CapableOf	MotivatedByGoal	
CausesDesire	HasProperty	
HasSubevent	MotivatedByGoal	HasProperty
HasProperty	Causes	
[老公][工作] 是為了 [有錢]		
[煩悶] 的 [工作] 會令人想要 [逃避]		
[有錢] 的時候會 [買房子]，是為了 [快樂] 的 [生活]		
但 [痛苦] 的 [房貸] 會帶來 [壓力]		

Table 3.3: Paragraph template

sentences in paragraphs is four or five, and each sentence has two or three relations. The first concept in paragraph template is a person, animal or something can conduct the actions (physical or mental) or be described. Some of the paragraph templates have backup sentence templates because concepts may not have corresponding connected concepts in some relations.

3.4.2 MCTS-based generated system

We combine coherence model and templates into our MCTS-based generated system. The flow chart of the generated system is in figure 3.7. User inputs an initial concept (subject), we first check whether it exists in ConceptNet or not. If not in ConceptNet, we substitute initial concept for its plesionyms or user can reinput it. And we adopt MCTS to simulate different combinations of concepts and evaluate coherence of generated paragraphs by the coherence model. We paraphrase the paragraphs generated by MCTS with plesionyms and pronouns. The replacement of pronouns can use Cilin (category A is human and category Bi is animal) or Named Entity Recognition (NER) to tag subjects as 他 or 牠. Evaluate those

paraphrased paragraphs by the coherence model again. Lastly, select one of the best results as our final generated paragraphs.

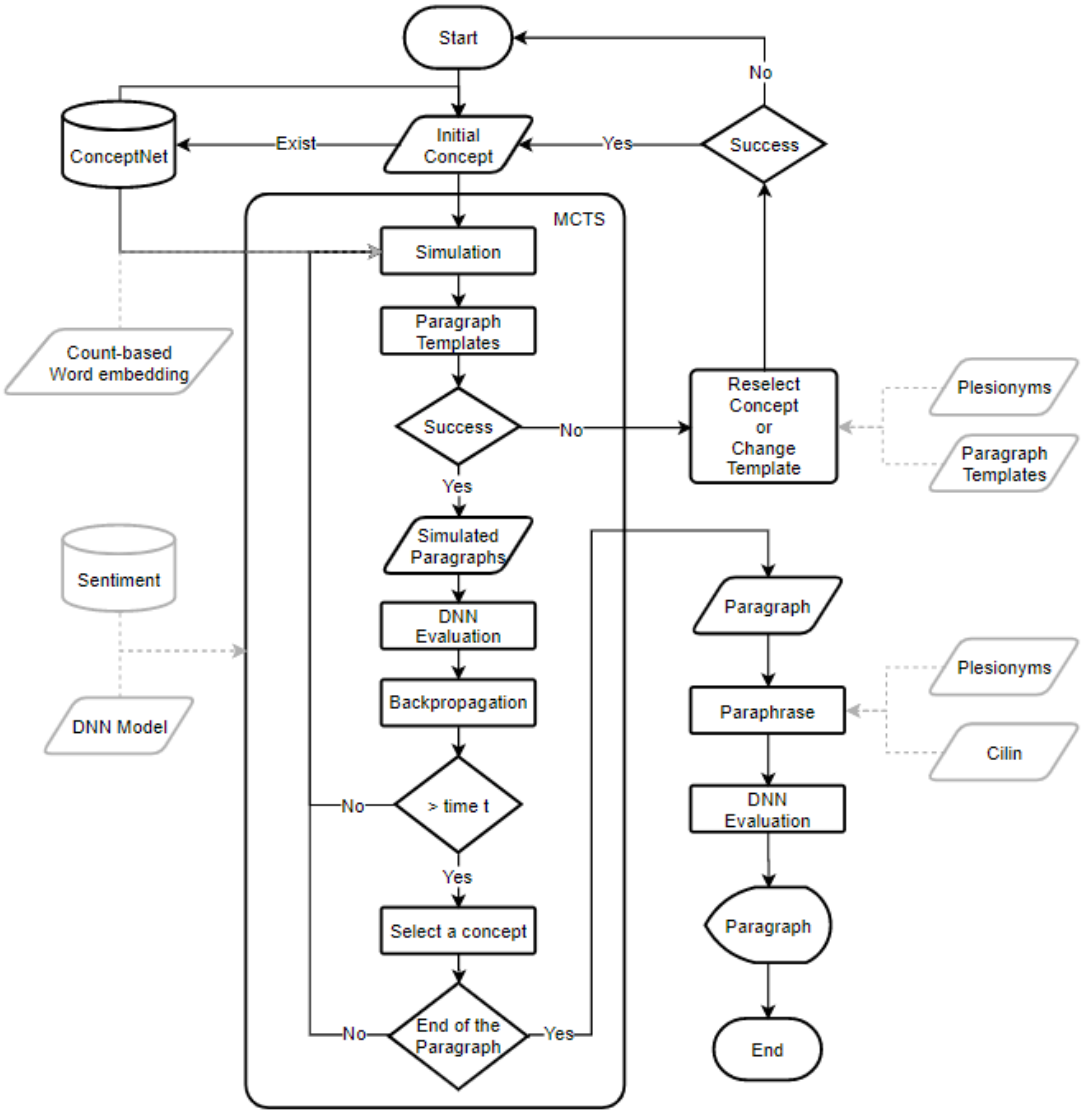

Figure 3.7: System flowchart.

MCTS

As we discussed in section 2.4.2, the search space is too large to find optimal or suboptimal paths by exhaustive search. MCTS is a heuristic algorithm which can find suboptimal paths in limited resources or time. An optimal decision algorithm is unnecessary in NLG tasks. Although it will find the most coherent paragraph,

it only finds single or few paragraphs as long as the initial concept has not been changed. We can view the problem of generating paragraphs as a decision tree search problem. MCTS builds an asymmetric tree instead of unfolding all possible paths. It can search coherent paragraphs in an extremely large tree. There are four steps in each round of MCTS which are selection, expansion, simulation, backpropagation. We will introduce these four steps in following descriptions.

Selection

Starting from a root node which is user's initial input concept. The tree continue traversing until reaching a leaf node. Each node in paths is concept that connected in ConceptNet. The selection in MCTS is a multi-armed bandit problem [42]. The gamblers have to evaluate the tradeoff between exploration and exploitation. They hope to maximize expected gain in limited resources. The simple selection strategy is ϵ-first. It is a pure random exploration in the first ϵT trials, followed by pure exploitation in the remaining times (1-ϵ)T. There are also other selection strategies, such as ϵ-greedy or ϵn-greedy. All of these strategies have some drawbacks. Either it has the same probability to choose good or bad arms in exploration phase, or the same expected gain arms have the same probability to be chose without considering the selected times in exploitation phase. Kocsis et al., [90] used Upper Confidence Bounds (UCB1) in MCTS called Upper Confidence bounds applied to Trees (UCT) as a tree policy in selection phase. In formula 3.11, $\frac{Q(v_i)}{N(v_i)}$ is the exploitation term. The child node v_i with higher score Q has higher probability to be selected. The tree should explore other nodes rather than selecting promising child nodes repeatedly. $\alpha\sqrt{\frac{2\ln N(V)}{N(v_i)}}$ is the exploration

term. If the visited times of node v_i is lower than other nodes, the exploration term becomes larger. Therefore, the tree can exploit promising nodes as well as exploring potential nodes

$$UCB1 = \frac{Q(v_i)}{N(v_i)} + \alpha\sqrt{\frac{2\ln N(V)}{N(v_i)}} \tag{3.11}$$

$Q(v_i)$: the reward value of node v_i
$N(v_i)$: the visited times of node v_i
$N(V)$: the visited times of root node V
α : a constant parameter in range 0 to 1

As long as the optimal branch or suboptimal branches haven't been discovered in early stages, the previous statistics are very misleading. It will spend too much time to explore those non-optimal branches. Consequently, besides the original UCT, we consider the sentiment of nodes. There are lots of unvisited child nodes in each round of selection because of the large search space and limited time. In order to find suboptimal paths in limited time, the rank of child nodes which have the same sentiment polarity to target node (previous selected node in upper layers) will be moved up, i.e., they have higher priority to be selected. The ranks of the rest of nodes are according to the polarity of target node. If the polarity of target node is positive or negative, the second group of polarity followed by the first group is neutral. We hope the sentiment polarity of child node is the same as target node as more as possible. Therefore, we put the neutral nodes in the middle of positive and negative nodes to separate.

We use sentiment dictionary instead of training a sentiment classifier. Combine

polarities by simple rules in table 3.4 if words are segmented. The polarity of segmented words are combined from the left to the right side iteratively until the last segmented one. The first column is the left side, and the first row is the right side. E.g., 保護 (P) 弱小 (N) → P, 假裝 (N) 乖巧 (P) → N, 出手 (N) 打人 (N) → N. The polarity is opposite if privatives are contained in segmented words, e.g., 從未 (V) 違規 (N) → P. There are still some words don't match rules, but the number of exceptions is acceptable.

We combine NTUSD [91], ANTUSD [92], CVAW2.0 [93] and 情感詞彙本體 [94] as our sentiment dictionary . Some of the words are in multiple classes, it may be an ambiguous word. We correct this type of problem so that one word only has one polarity. Besides these resources, we also expand sentiment dictionary by plesionyms. The size of sentiment dictionary is 22124 positive words, 26554 negative words and 27503 neutral words.

	P	N	T
P	P	P	P
N	N	N	N
T	P	N	T
V	N	P	T

Table 3.4: Combination rules of sentiments.

P : Positive
N : Negative
T : Neutral
V : Privative

Lastly, we select from top N child nodes which have higher scores. The N value is not fixed, it is inappropriate to select fixed number of nodes regardless of the children size, hence, it has different values according to the number of children.

Expansion

If the current node is a leaf node, the tree expands possible child nodes and chooses one of them randomly to simulate. Because not every child node is coherent to previous selected nodes in upper layers, we calculate the cosine similarity between child nodes and upper layer nodes. Child nodes won't be expanded if they are OOV or the cosine similarity score is less than zero. In order to reduce the computational cost, we expand current node only if the ratio of valid child nodes to invalid ones is less than 0.3. The current node will be removed if it can't expand child nodes (no connected concept in ConceptNet) or it is an invalid node. The nodes in upper layers will be removed too if they are not terminal nodes and don't have any child node. The dead end path in a tree is unnecessary. We don't expand child nodes which have the same concept as selected nodes in upper layers to prevent repeated or redundant concepts in a generated paragraph.

Simulation

MCTS runs a simulation from current node to terminal node by a default policy which is selecting nodes randomly according to paragraph templates. Simulate randomly is quick and simple. It doesn't need any domain knowledge and it can cover different regions of search tree.

However, it needs more time to converge a better result. It needs more time to simulate in early stages (step $<$ Nth step) of simulation because there are lots of unvisited nodes, hence the simulated time is relatively low for each node. The inaccurate predictions caused by low simulated times may bias the result of

selection if simulating randomly. The impact of errors in early stages is bigger than late stages because the selected nodes in late stages are based on upper layer nodes in the same path. Most of the nodes can be simulated in late stages of simulation because the number of unvisited nodes is low. The predictions are more accurate than early stages as the visited times increasing. In order to prevent the inaccurate simulation in early stages, we simulate with domain knowledge by word embedding. We adopt two methods here.

- The concepts which are more related (higher cosine similarity score) to target concept have higher probability to be simulated. For example, I go to fast food restaurant to eat fried chicken. The food is healthy. It's unlikely to say that fried chicken is a healthy food. The slot would be more reasonable by calculating the relatedness to other concepts.

- If the simulated concept is the third node in a sentence, we use the first and the second concept to predict the third concept by word embedding. The third concept which is related to the first two concepts has higher probability to be simulated. This is similar to sentence cloze test. The participants are asked to fill in the empty slot given other words in a sentence, e.g., I went to __ to buy fast food. The slot may be McDonald's or other fast food restaurants.

The advantages of these two methods are that they can avoid simulating unreasonable moves to reduce the computational cost. It needs less time to converge a better result, and have more time to explore or to exploit other nodes. Increase the accuracy of simulation by increasing the visited times in finite time. The dis-

advantages of them are that they need to train a word embedding and may not simulate some potential nodes since the word embedding is not 100 % accurate. The scores of some nodes are zero or even negative since they may not related to target concept or first two concepts in a sentence.

In order not to miss every possible node, we transform the distribution of scores. We don't use the cosine similarity scores or the probability of predicted concepts directly. First, we normalize the scores so that the summation is 1, and find a minimal value. Add the minimal value ($\log 0$ is undefined) and apply log transformation to make right-skewed distribution data (data concentrate on left side) more normal distribution. We then shift the data to positive by adding minimal value + 0.1 to avoid zero scores. Lastly, normalize the distribution again after log transformation. We don't want the highest related nodes to be simulated over and over, and ignore the low related nodes. As a result, we decrease the probability of maximal value and increase the probability of minimal value. It becomes more normal distribution than the original one.

The reward value of terminal node is not a simple binary representation which is winning (1) or losing (0) of a game. It's hard and complicated to evaluate how coherent a paragraph is. We use our coherence model instead of binary representation which is introduced in section 3.3. The coherent scores of terminal nodes will be evaluated at each time of simulation.

If concepts don't have any connected concepts, we use backup sentence templates to replace the current ones. If concepts still can't find any connected concepts, we change the paragraph template or initial concept by plesionyms. All of the information (parent, children, value, relation, reward value and visited times)

stored in each node will be reset, and restart the new MCTS round.

Backpropagation

Update $Q'(v_i)$ to simulated reward value $Q(v_i)$ of each node from current to root node in the traversed path. The visited times of nodes $N(v_i)$ in the path is incremented by 1.

$$s_i^{update} = \frac{Q(v_i) + Q'(v_i)}{N(v_i) + 1} \tag{3.12}$$

We reuse the subtree information in past time-steps of searching at subsequent time-steps. A node will be selected after a round of MCTS. This node is the new root node for the next round. We retain the subtree statistics below this new root and discard its parent node and the remainder of other sibling subtrees.