UNIVERSITY OF CAMBRIDGE

MPhil in Machine Learning and, Machine Intelligence 2018-2019

Module Coursework Feedback

Module Title: Probabilistic Automata

Module Code: MLMI3

Candidate Number: F606F

Coursework Number: 1

I confirm that this piece of work is my own unaided effort and adheres to the Department of Engineering's guidelines on plagiarism. √

Date Marked: Click here to enter a date. Marker's Name(s): Click here to enter text.

Marker's Comments:

This piece of work has been completed to the following standard *(Please circle as appropriate)*:

	Distinction			**Pass**			**Fail (C+ - marginal fail)**		
Overall assessment (circle grade)	**Outstanding**	**A+**	**A**	**A-**	**B+**	**B**	**C+**	**C**	**Unsatisfactory**
Guideline mark (%)	90-100	80-89	75-79	70-74	65-69	60-64	**55-59**	**50-54**	**0-49**
Penalties	**10% of mark for each day, or part day, late (Sunday excluded).**								

The assignment grades are given **for information only**; results are provisional and are subject to confirmation at the Final Examiners Meeting and by the Department of Engineering Degree Committee.

1. Practical Exercise Part 1

Question 1

Assume that the alphabet $L = \{a, b, c, A, B, C, < space >\}$, a word is defined as a string of more than 1 letter from L excluding $< space >$. Any string can be classified as the word when it is formed by any sequence of alphabets from L but any capitalized letter can't be at any position of the string other than at the beginning unless it is required. A capitalized word is defined as a word which has its initial letter uppercase and the remaining lowercase. Any capitalized letter is not considered as equivalent as the letter itself.

Based on the assumptions above, the following tasks are completed.

a.

The automaton A which accepts any letter in L (including $< space >$) is created. It accepts any letter and space from start state 0 to end state 1. For simplicity, each arc has no weight.

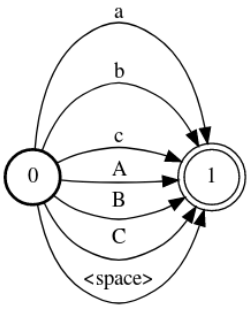

Figure 1: Automaton A

b.

The automaton B is created to accept a single space $< space >$.

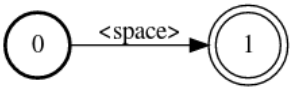

Figure 2: Automaton B

c.

The automaton C shown below accepts any capitalized word of any length. The transitions from start state 0 to state 1 ensure the first letter to be capitalized. Those from state 1 to end state 2 enable the assumption that a word must be a string with more than 1 letter. The self-transitions at end state 2 allow the word to have infinite length.

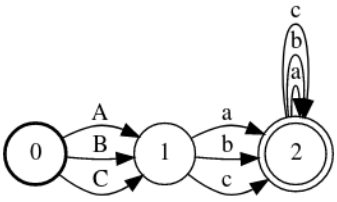

Figure 3: Automaton C

d.

The first automaton $D1$ is created to accept any word beginning with the letter a.

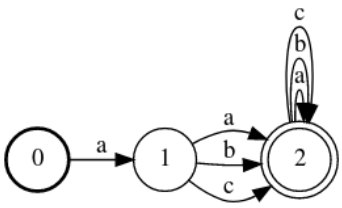

Figure 4: Automaton D1

The second automaton $D2$ is created to accept any word that has the letter a at any position of the string other than the beginning.

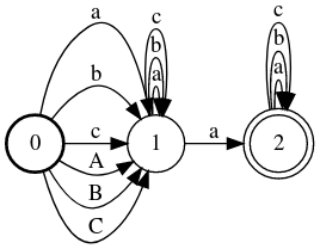

Figure 5: Automaton D2

The resulting automaton D which can accept any word containing letter a is created via the union of $D1$ and $D2$. The machine D is formed by the following operation,

$$D = D1 \cup D2$$

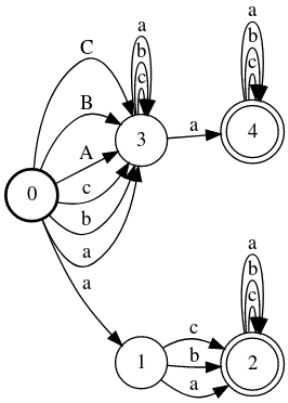

Figure 6: Automaton D

Question 2

The standard set of operations O used in this question is defined as epsilon removal, determinization and minimization. The notation $*$ marks the Kleene Closure operation. The notation Π represents the determinization operation, the notation $-\epsilon$ denotes the epsilon removal and min represents the minimization process.

a.

The machine C is firstly concatenated with B to form another machine which is then undergone the Kleene Closure operation to accepts one or more capitalized words followed by spaces.

The second branch is equal to the machine B which accepts zero capitalized word with a space.

Lastly, the union of these two machines form the machine $A2$ which accepts zero or more capitalized words followed by spaces.

$$A2 = (C|B)^* \cup B$$

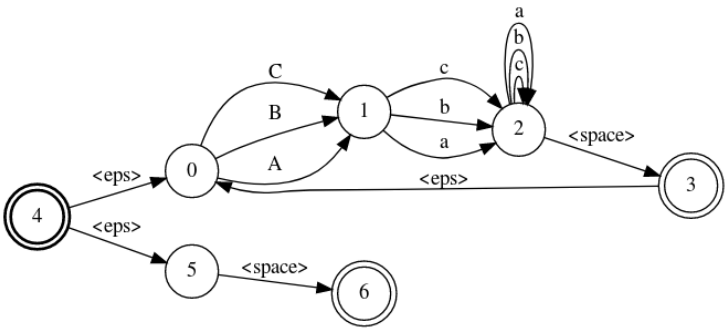

Figure 7: Automaton A2

The automaton $A2$ has 14 arcs and 7 states before the operations.

$$A2_{after} = O(A2) = min[\prod((C|B)^* \cup B)_{-\epsilon}]$$

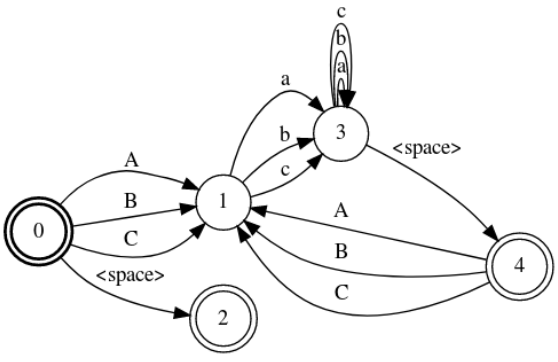

Figure 8: Automaton A2 after operations

After the operations, it now has 14 arcs and 5 states.

b.

The following automaton in Figure 9 is created via the reverse operation of C, i.e. $\overline{C}$. It accepts any word ending in a capitalized letter.

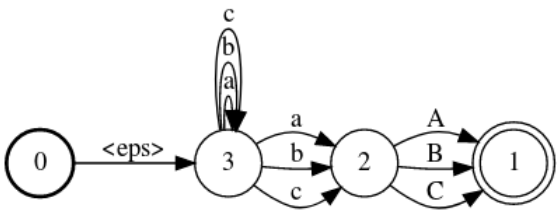

Figure 9: Automaton $\overline{C}$

C is then concatenated with $\overline{C}$ to form a machine which accepts any word beginning and ending in a capitalized letter.

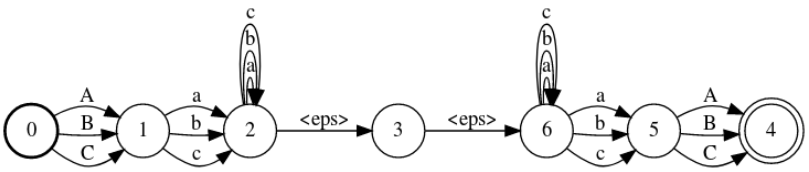

Figure 10: Automaton formed by $C|\overline{C}$

The resulting $B2$ machine is then generated by the union of C, $\overline{C}$ and $C|\overline{C}$ to accept any word beginning or ending in a capitalized word.

$$B2 = C \cup \overline{C} \cup \left(C|\overline{C}\right)$$

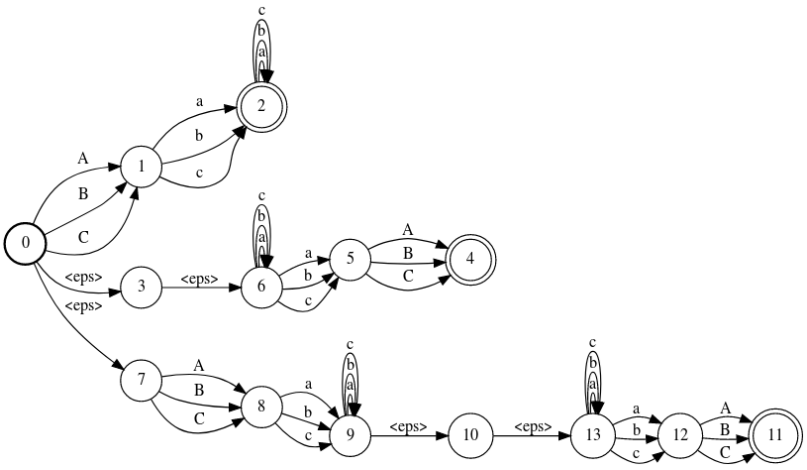

Figure 11: Automaton B2

Before the operations, it has 41 arcs and 14 states.

$$B2_{after} = O(B2)$$

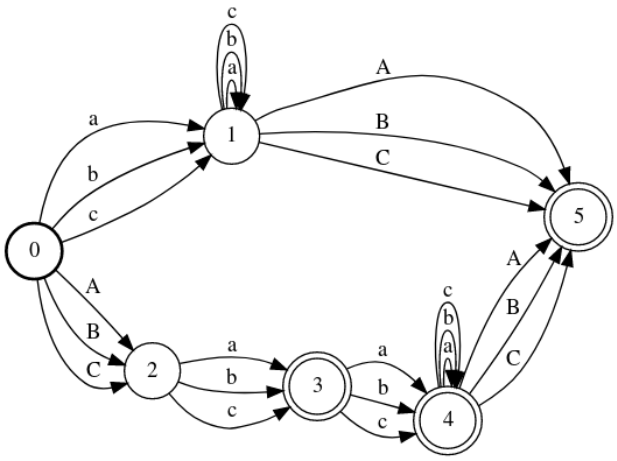

Figure 12: Automaton B2 after operations

After the operations, it has 24 arcs and 6 states.

c.

The machine $C2$ is the intersection between machines C and $D2$. It accepts any capitalized word with the letter a.

$$C2 = C \cap D2$$

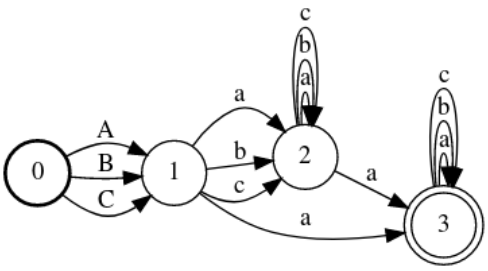

Figure 13: Automaton C2

The number of arcs and states are 14 and 4 respectively.

$$C2_{after} = O(C2)$$

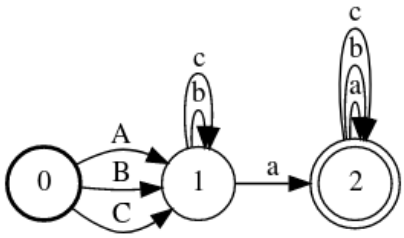

Figure 14: Automaton C2 after operations

The number of arcs and states are 9 and 3 respectively.

d.

The automaton $D3$ which accepts the capitalized word with the letter a is created through the difference operation, i.e. $D3 = C - C2$.

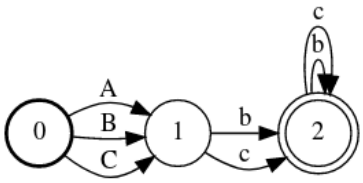

Figure 15: Automaton D3

From Question 2c, the machine $C2$ can be used to accept any capitalized word with the letter a.

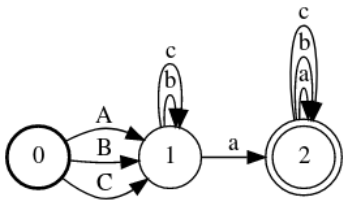

Figure 16: Automaton C2

To create a machine which accepts any non-capitalized word without the letter a, the followings have been done.

An automaton $L1$ in Figure 17 which accepts any letter excluding <space> is created via $A - B$.

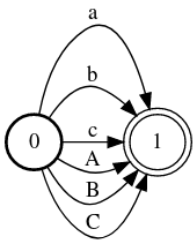

Figure 17: Automaton A-B

Then the 'shortestpath' operation operates on $L1$ to create machine which only accepts a single letter. For instance, since all arcs are weighted equally and the letter a is on the outermost arc, this operation firstly creates a machine 'as' which only accepts a letter a.

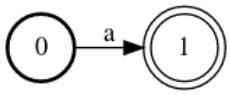

Figure 18: Automaton as

Similarly, any machine which accepts any single letter can be created via difference followed by shortestpath operation. For example, the next machine 'bs' which only accepts a letter b can be done by,

$$bs = shortestpath(L1 - as)$$

In general, any machine which only accepts a single element (letter) $l_n \subset L$ can be created, where n the index of the alphabet in L,

$$l_n = shortestpath(A_{+k} - l_k)$$

After creating automata in which each only accepts a single distinct letter, an automaton can be created to accept any letters intended. Since a machine which does not accept capitalized letters and the letter a is wanted, an automaton $D4$ which only accepts b and c is created by $L1 - as - As - Bs - Cs$.

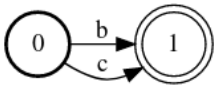

Figure 19: Automaton D4

Finally, the automaton which accepts any word without capitalzed and the letter a is created via $(L1 - as - As - Bs - Cs)^*_{-\epsilon}$

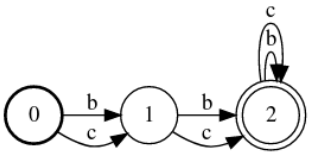

Figure 20: Automaton $D4|D4^*_{-\epsilon}$

The overall machine $D5$ is generated by,

$$D5 = (C - C2) \cup C \cup (L1 - as - As - Bs - Cs)^*_{-\epsilon}$$

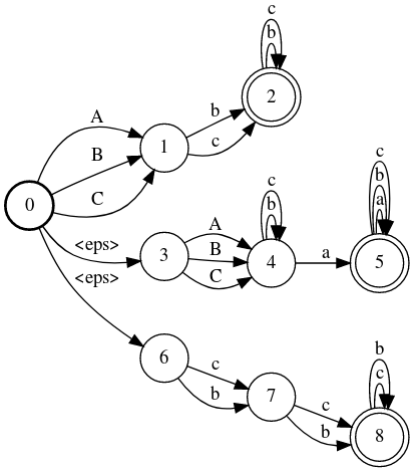

Figure 21: Automaton D5

Before the operations, the number of arcs and states are 23 and 9 respectively.

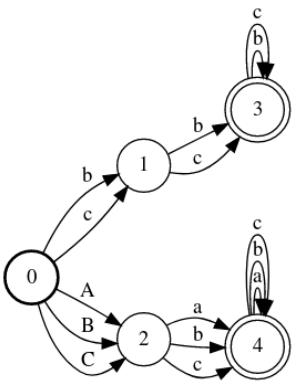

Figure 22: Automaton D5 after operations

After the operations, the number of arcs and states are 15 and 5 respectively.

e.

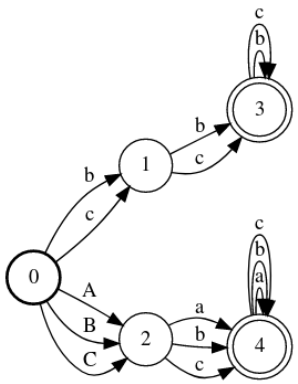

Figure 23: Automaton D5 built without union

Based on the machine C, the machine $D4$ from Figure 20 is added to it manually through 'add_arc' operation to avoid the union operation.

Before and after the operations, the number of arcs and states are 15 and 5 respectively.

Question 3

a.

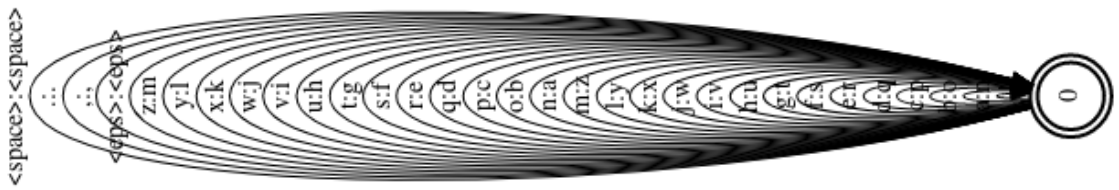

Figure 24: FST T13

A transducer $T13$ is created to encode each letter to another letter according to the table below.

Table 1

Input	a	b	c	d	e	f	g	h	i	j	k	l	m
Output	n	o	p	q	r	s	t	u	v	w	x	y	z
Input	n	o	p	q	r	s	t	u	v	w	x	y	z
Output	a	b	c	d	e	f	g	h	i	j	k	l	m
Input	<space>		,	.	<eps>								
Output	<space>		,	.	<eps>								

b.

An FSA M is created to accept the string ‘my secret message’.

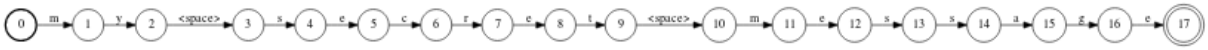

Figure 25: Automaton M

After the composition with $T13$, It is projected to the output and undergone epsilon removal to give the encoded message ‘zl<space>frperg<space>zrffntr’. It gives the following machine E,

$$E = \Pi_2[M \circ T13]$$

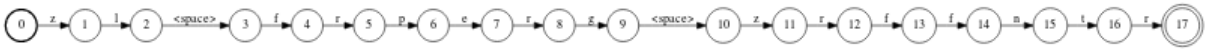

Figure 26: Automaton E

To decode the message, E is composed with $T13$ again. It is undergone with the output projection and epsilon removal to give the machine M.

$$M = \Pi_2[E \circ T13]$$

c.

A decoding transducer T is built to decode a letter to either two letters. The transformation process is listed as follows,

Table 2

Input	a	b	c	d	e	f	g	h	i	j	k	l	m
Output1	n	o	p	q	r	s	t	u	v	w	x	y	z
Output2	q	r	s	t	u	v	w	x	y	z	a	b	c
Input	n	o	p	q	r	s	t	u	v	w	x	y	z
Output1	a	b	c	d	e	f	g	h	i	j	k	l	m
Output2	d	e	f	g	h	i	j	k	l	m	n	o	p
Input	<space>		,	.	<eps>								
Output	<space>		,	.	<eps>								

Figure 27: Automaton T

After the decoder is built, the FSA $M1$ which accepts the message is composed with T. The result is then projected to the output to give R.

$$R = \Pi_2[M1 \circ T]$$

Before the operations, the number of arcs and states are 1016 and 365 respectively.

After the operations, the number of arcs and states becomes 649 and 366 respectively.

The number of distinct strings it can represent is $3.107 \text{x} 10^{85}$.

d.

To find the original text, machine R is composed with the language model LM, and undergone with the standard set of operations (epsilon removal, determinization and minimization) to give the machine $R1$.

$$R1 = O(R \circ LM)$$

The original text was given by the machine $R1$,

Whether I shall turn out to be the hero of my own life, or whether that station will be held by anybody else, these pages must show. To begin my life with the beginning of my life, I record that I was born as I have been informed and believe on a Friday, at twelve oclock at night. It was remarked that the clock began to strike and I began to cry, simultaneously.

e.

The second decoding transducer $T2$ is built to map all the punctuation marks to themselves in state 0. And it maps each letter to itself in state 0 and to other letters from state 0 to n, and maps other symbols back to itself from state n to 0. It is illustrated in Figure 28.

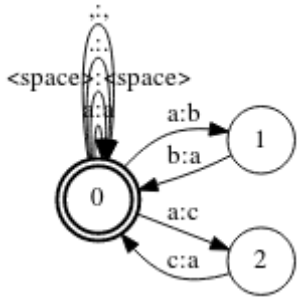

Figure 28: Part of Automaton T2

To extract the original text, the message $M2$ is undergone with the operation arcsort. Then it is composed with $T13$ and undergone the epsilon removal. Then it is composed with $T2$ and projected to the output. Finally, it is undergone with the standard set of operations to give the final machine $R2$.

$$R2 = O(\Pi_2[[M2 \circ T13]_{-\epsilon} \circ T2])$$

The original text was,

I don't know how long we were going, and to this hour I know as little where we went. Perhaps it was near Guildford. Perhaps some arabiannight magician, and shut it up for ever when we came away. It was a green spot, on a hill carpeted with soft turf. There were shady trees, and heather, and, as far as the eye could see, rich landscape.

Question 4

The number of distinct paths in the acceptor is approximately 487232.

Steps to count the number of distinct paths,

I. Remove the weight of the WFSA.
II. Convert the arc weight to log.
III. Compute the shortest distance.
IV. Convert the shortest distance by $exp(-shortestdistane)$.

The first step removes all the arc weight and sets to 1, each path is then weighted equally. The second step converts the machine to log-semiring, so all weight is now set to $log(1) = 0$.

So, the shortest distance operation adds up all the weights of all the shortest paths by the following equation,

$$\oplus_{\mathrm{i}}\, k_i = -log\left(\sum_i e^{k_i}\right)$$

, where k_i is the weight of each path i.

But since each of them is weighted by 0, the operation counts all the possible paths by adding $e^{k_i} = e^0 = 1$.

The returned value is then $-log(N)$, where N is the number of paths. Therefore, $exp(-[-log(N)]) = N$. It returns the number of distinct paths.

2. Practical Exercise Part 2

Question 1: Edit Transducer

a.

Define the file 'wmap/devtest.wmap' is the word set W, it contains every word used in this practical excluding white space ' ' , separator '|' and separators '||'.

Define the file 'editoptions/editoptions.devtest' is the word set E, and it contains a function F that maps the input string x to all its edit options set Y, where $x \in Y$.

$$Y = F(x)$$

If $y \in Y$ and $y \neq$ x,

$$F(y) = \emptyset$$

The relationship between W and E can be stated as follows,

$$E \cap W \subset W$$

$$E - E \cap W = \{' ', '||'\}$$

$$E^* - E^+ = \epsilon$$

Define the correction edit tag set as $C = \{< corr >, <\backslash corr >\}$.

Thus,

$$L_{T_1} = \{x | x \in (E \cap W - C) and\ x \notin (F(x) - x)\}$$

$$L_{T_2} = \{y | y \in E \cap W \cup (E^* - E^+) and\ y \in F(L_{T_1})\}^+$$

b.

Assume that S_{1198} is the set of words in the 1198th sentence of the CONLL2013 set, the input language (including the punctuation) can be expressed as,

$$L_{T_1} = \{x | x \in S_{1198} \cup (E^* - E^+) and\ \text{x} \notin (\text{E} - \text{E} \cap \text{W})\}^+$$

The output language can be expressed as,

$$L_{T_2} = \{y | y \in F(S_{1198}) \cup Candy \notin (E - E \cap W)\}^+$$

c.

After the output projection, the output language from question 1b is extracted to form the language of the acceptor $L_{T_2} \Rightarrow L_T$, where L_T the language of the acceptor.

The language of the acceptor is $L_T = \{y | y \in F(S_{1198}) \cup Candy \notin (E - E \cap W)\}^+$.

Question 2: Edit Penalties and Processing Speed

a.

The cost of a path p through the machine E:

$$[[W_n]](i(p):o(p)) = \lambda\#_{corr}(p:i(p):o(p)) + P_{bigram}(p:i(p):o(p))$$

b.

Consider two paths p_1 and p_2 through W_n, the path with the minimized weight is chosen. Thus, the decision rule can be expressed as follows,

$$\text{argmin}_{\text{p}}[[\text{W}_\text{n}]](\text{i(p)}:\text{o(p)})$$

Assume that p_1 is the path with no correction and p_2 is the path with correction, the cost of p_1 is given by,

$$[[W_n]](p_1) = P_{bigram}(p_1)$$

The cost of p_2 is given by,

$$[[W_n]](p_2) = \lambda\#(p_2) + P_{bigram}(p_2)$$

If $\lambda < [[W_n]](p_1) - [[W_n]](p_2)$, it favours the path with corrections and p_2 is chosen. If $\lambda \geq [[W_n]](p_1) - [[W_n]](p_2)$, it favours the path without correction and p_1 is chosen.

c.

The procedure described in section 4 of B&B'18 chooses to correct the most serious errors first, in terms of language model probability increase. It minimizes false positives by setting a threshold such that a candidate correction must improve the average token probability of the sentence before it is applied.

And the method from this question chooses to correct the errors which can lead to the minimum weight of the path.

The former always re-estimates the log-probability of the sentence after each correction, but the latter does not. The former has a threshold for the change, so it is more accurate, but the latter does not. The former sets the correction priorities, but the latter does not.

d.

Modified construction procedure for the automata E_n is listed as follows,

I. The LMmap transducer T_{LMmap} is determinized.
II. The sentence WFSA O_n is composed with the edit transducer T_{edit}.
III. It is projected on its output.
IV. It is determinized and minimized. (modified)
V. It is composed with the LMmap transducer T_{LMmap}.
VI. It is composed with the bigram language model WFSA G_2.
VII. It is undergone with epsilon removal, determinization and minimization. (modified)

e.

Several trials are done for performing time measurements on the first 500 sentences of the CONLL2013 set. The overall time for final GEC operations on 500 sentences is recorded in the table.

Table 3

Trial	Before(s)	After(s)
1	2.913	1.976
2	2.778	1.797
3	2.690	1.749
4	2.637	1.774
Average	2.755	1.824

The modified procedure for the automata E_n does improve the speed of the final GEC operations.

Question 3: Oracle Sentence Error Rate

a.

It is found that the number of states after intersection depends on the target(s) the FST intersects with, therefore, the Oracle Sentence Error Rate (OSER) should also depend on the intersected target(s). To compute OSER as the function of number of reference corrections, the average of OSERs of the combinations of the reference sets is computed.

For each n reference corrections, where n is the number of reference corrections, there are C_n^4 combinations of reference sets which lead to the following tables.

Table 4

1 reference correction combinations	0	1	2	3
Oracle Sentence Error Rate	0.837	0.850	0.817	0.759
Average	0.816			
Total	0.642			

Table 5

2 reference corrections combinations	0, 1	0, 2	0, 3	1, 2	1, 3	2, 3
Oracle Sentence Error Rate	0.779	0.755	0.715	0.761	0.711	0.695
Average	0.736					
Total	0.642					

Table 6

3 reference corrections combinations	0,1,2	1,2,3	0,2,3	0,3,1
Oracle Sentence Error Rate	0.718	0.663	0.664	0.682
Average	0.682			
Total	0.642			

Table 7

4 reference corrections	0,1,2,3
Oracle Sentence Error Rate	0.642

The Oracle Sentence Error Rate for the JFLEG DEV set decreases with the increase of the number of reference corrections.

It is interesting to note that if the union is done between the lists of sentence errors for each combination, the total error rate must equal to the error rate for 4 reference corrections.

b.

The m2 score includes the precision value, the recall value and the $F_{0.5}$ score, they are given by the following equations,

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_{\beta} = (1 + \beta^2)\frac{PR}{\beta^2 P + R}$$

, where $\beta = 0.5$, TP the number of true positive, FP the number of false positive, FN the number of false negative, P the precision and R the recall.

Increasing the number of references should help to increase the number of TP but decrease the number FP and FN. Thus, all the scores should be increased with the number of references.

Not all the increase or decrease should be trusted as the changes depend on which references are undergone union and which references the original one intersects with. The changes can be trusted if the average is taken.

Question 4

a.

Table 8

		CONLL 2013	CONLL 2014		JFLEG DEV	JFLEG TEST
N-Gram Order	Edit Cost	$F_{0.5}$	$F_{0.5}$	Edit Cost	$F_{0.5}$	$F_{0.5}$
1	2.5	0.1224	0.1829	6.0	0.4617	0.5136
2	2.5	0.2181	0.2791	6.0	0.5466	0.5939
3	2.5	0.2540	0.3329	6.0	0.5697	0.6039
4	2.5	0.2530	0.3261	6.0	0.5654	0.5928

The $F_{0.5}$ values for all datasets increase with the N-Gram Order until N=3. Those values begin to decrease at 4-gram.

b.

Precision is denoted as 'P' value and Recall is denoted as 'R' value.

Table 9

		CONLL 2013			CONLL 2014 (short)				JFLEG DEV			JFLEG TEST		
N-Gram Order	Edit Cost	*P*	*R*	$F_{0.5}$	*P*	*R*	$F_{0.5}$	Edit Cost	*P*	*R*	$F_{0.5}$	*P*	*R*	$F_{0.5}$
3	1.0	0.273	0.174	0.245	0.318	0.263	0.305	4.0	0.686	0.321	0.558	0.705	0.348	0.585
3	2.0	0.313	0.145	0.255	0.372	0.220	0.327	5.0	0.724	0.302	0.566	0.749	0.332	0.598
3	2.5	0.333	0.130	0.254	0.401	0.196	0.332	6.0	0.758	0.286	0.570	0.785	0.314	0.604
3	3.0	0.365	0.117	0.256	0.431	0.181	0.338	7.0	0.790	0.267	0.567	0.806	0.297	0.600
3	4.0	0.405	0.085	0.231	0.494	0.147	0.336	8.0	0.811	0.252	0.561	0.832	0.279	0.596
3	5.0	0.466	0.066	0.211	0.579	0.123	0.332	9.0	0.824	0.237	0.551	0.846	0.266	0.589

With the fixed N-Gram Order – 3, the precision values increase, and the recall values decrease for all datasets while the edit-cost is increasing.

For the CONLL set, the $F_{0.5}$ score begins to decrease when the edit cost is equal to 4.0.

For the JFLEG set, the $F_{0.5}$ score starts to decrease when the edit cost is equal to 3.0.

Question 5

Table 10

	CONLL 2013			CONLL 2014 (short)			JFLEG DEV			JFLEG TEST		
N-Gram	*P*	*R*	$F_{0.5}$	*P*	*R*	$F_{0.5}$	*P*	*R*	$F_{0.5}$	*P*	*R*	$F_{0.5}$
5	0.4152	0.1331	0.2916	0.4870	0.2150	0.3887	0.7473	0.3136	0.5854	0.7664	0.3435	0.6150

The steps to complete the table is described as follows,

I. Compose each sentence acceptor O_n from the dataset with tedit transducer T_{edit}.
II. Project the output Π_2.
III. Compose it with the LMmap transducer T_{LMmap}.
IV. Compose it with the 5-gram language model G_5.
V. Compose the correction penalty A_λ with the result from step IV.
VI. Project the input and compose it with word to character FST T_{wtc}.
VII. Project the output Π_2.
VIII. Operate epsilon removal, determinization and minimization.
IX. Configure and run SGNMT to get the m2 score.

Steps I – IV can be expressed into the following equation,

$$E_n = \Pi_2[O_n \circ T_{edit}] \circ T_{LMmap} \circ G_5$$

Step V composes the correction penalty.

$$W_n = A_\lambda \circ E_n$$

Steps VI – VII is given by the following equation,

$$C_n = \Pi_2[\Pi_1[W_n] \circ T_{wtc}]$$

The cost of the character sequence then becomes

$$\big[[C_n]\big](c) = \lambda \times \#_{corr}(w') + P_5(w')$$

For a character level sequence c, the SGNMT decoder assigns the score by

$$w_{t2t} \times P_{t2t}(c) + w_{fst} \times \big[[C_n]\big](c)$$

$$= w_{t2t} \times P_{t2t}(c) + w_{fst} \times \big(\lambda \times \#_{corr}(w') + P_5(w')\big)$$
$$= w_{t2t} \times P_{t2t}(c) + w_{fst} \times \lambda \times \#_{corr}(w') + w_{fst} \times P_5(w')$$
$$= w_{t2t} \times P_{t2t}(c) + w_{corr} \times \#_{corr}(w') + w_{LM} \times P_5(w')$$

, where $w_{corr} = \lambda \times w_{fst}$, $w_{LM} = w_{fst}$, λ the edit cost and w_{LM} the scale factor of the language model.

Therefore, for the CONLL sets, when $w_{t2t} = 0.625$, $w_{corr} = 1.875$ and $w_{LM} = 0.25$, w_{fst} should be 0.25 and λ should be $\frac{1.875}{0.25} = 7.5$.

Similarly, for the JFLEG sets, when $w_{t2t} = 0.1875$, $w_{corr} = 1.5$ and $w_{LM} = 0.171875$, $w_{fst} = 0.171875$ and $\lambda = \frac{1.5}{0.171875} \approx 8.73$.

These calculated numbers should correspond to the values of 'predictor_weights: w_{t2t}, w_{fst}' in the configuration files 'char.translm_5gfst_4_XXXX' where 'XXXX' is the dataset name which includes 'conll2013', 'conll2014.short', 'jfleg_dev' and 'jfleg_test'.

Other Investigations into GEC architectures

When completing the tasks above, several mistakes are made in different sessions. These mistakes accidentally reveal the importance of each transducer to the GEC architectures.

The LMmap transducer initially built does not properly consider the whole list of edit tags, including <spell>, <infl>, etc., the m2 scorer then fails to give scores to the text file created, which results in the errors with all 0s for precision, recall and $F_{0.5}$ score. This might be because when those tags are not removed, the later models do not have the function to remove them, this makes the text file created eventually looks completely different from the reference text. It then results in the failure of scoring.

After re-running with the modified LMmap transducer, the results remain the same. And it is found that the n-gram edit acceptor for each sentence and the created text file for each set need to be deleted before re-running. And finally, some results are obtained and listed in the following tables.

Table 11

		CONLL 2014		JFLEG TEST	
N-Gram Order	Edit Cost	$F0.5$		Edit Cost	$F_{0.5}$
1	2.5	0.1820		6.0	0.5136
2	2.5	0.2767		6.0	0.5856
3	2.5	0.3291		6.0	0.5948
4	2.5	0.3226		6.0	0.5855

		CONLL 2014 (short)				JFLEG TEST		
N-Gram Order	Edit Cost	P	R	$F_{0.5}$	Edit Cost	P	R	$F_{0.5}$
3	1.0	0.315	0.261	0.303	4.0	0.696	0.340	0.576
3	2.0	0.369	0.218	0.324	5.0	0.742	0.322	0.589
3	2.5	0.396	0.194	0.328	6.0	0.778	0.307	0.595
3	3.0	0.426	0.179	0.333	7.0	0.800	0.291	0.592
3	4.0	0.486	0.145	0.330	8.0	0.826	0.273	0.587
3	5.0	0.572	0.120	0.327	9.0	0.841	0.259	0.581

It is found that the filled numbers do not match those from Table 8 and 9. Especially, for those with non-unigram, they deviate much more. And it shows the mistake made in the initial tedit transducer.

In the edit option map, some words with the length of 1 string are found to transform into two or more words. But in the original tedit transducer created, only the mappings from one word to one word are included. This explains why the unigram numbers in Table 11 match those in Table 8 and 9, but with N-gram order greater than 1, those numbers begin to differ. This shows the tedit transducer construction affects the results of n-gram model if those mappings between n-words are not properly accepted.