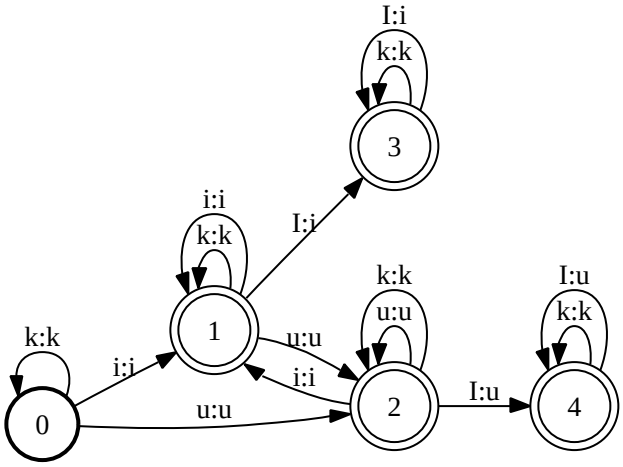
CMPT 413: Sample Questions for Midterm

The actual midterm will be shorter. These questions are to help you prepare for the midterm.

(1) Consider the following finite-state transducer (FST) γ, where we assume that *u* and *i* are the only vowels in this particular natural language and that *k* is used as a placeholder for any consonant. *I* is an abstract vowel that appears in the input language and depending on the context is mapped to either *u* or *i* in the output language.



a. The *input* language of the above FST can be written as the regular expression:

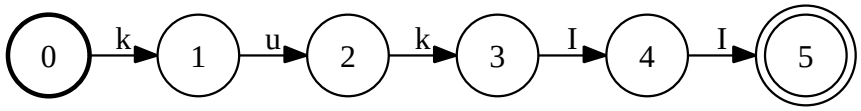
`(k* (u|i))+ k* (k*I)* k*`

Does the string *kiiukI* belong to the input language?

Answer: 1pt: Yes.

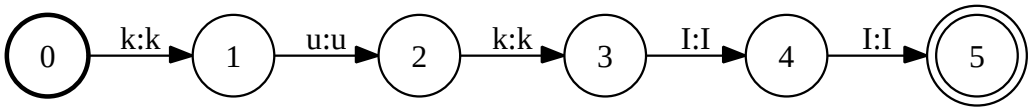
b. Provide a finite-state machine (FSM) *f* that accepts the string *kukII* and nothing else.

Answer: 1pt:



c. *Id* is a function that takes a FSM as input and produces a FST that accepts pairs of strings, where each string accepted by the FSM is paired with itself. Provide *Id*(*f*).

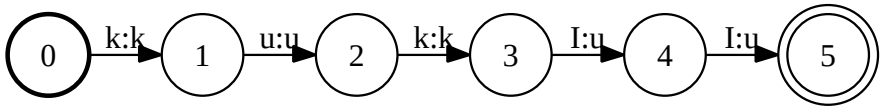
Answer: 2pt:



d. Provide $Id(f) \circ \gamma$, the composition of $Id(f)$ with the FST γ given above.

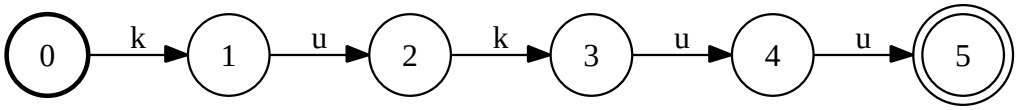
Answer: 6pt:

```
(0,0) (1,0) k:k
(1,0) (2,2) u:u
(2,2) (3,2) k:k
(3,2) (4,4) I:u
(4,4) (5,4) I:u
```



e. Provide the FSM $\pi_2(Id(f) \circ \gamma)$, where π_2 projects the output language of a FST to provide an FSM.

Answer: 2pt:



f. Provide an alternate name used to denote steps (1b) through (1e) that produced $\pi_2(Id(f) \circ \gamma)$.

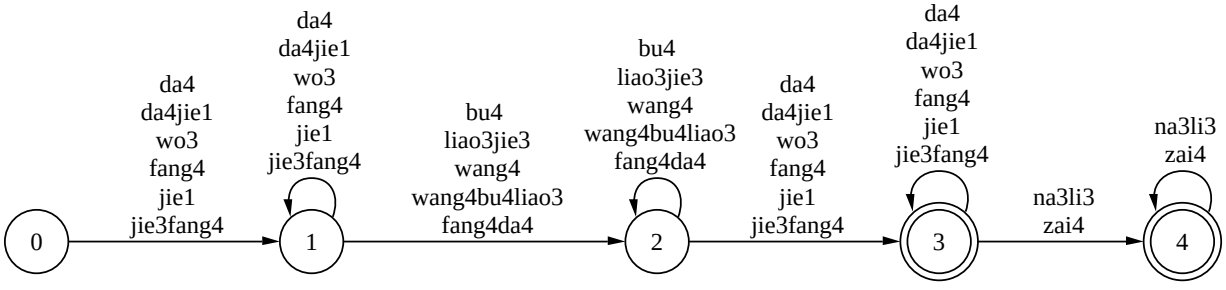
Answer: 2pt: $\pi_2(Id(f) \circ \gamma) = transduce(\gamma, kukII)$

g. Provide two rewrite rules of the type $\alpha \rightarrow \beta / \lambda \_ \rho$ that are together equivalent to the FST γ. Mention whether they are *obligatory* or *optional* and if they are to be applied *left to right* or *right to left*, and whether they are used *iteratively* or *simultaneously*.

Answer: 6pt: These two obligatory rewrite rules are left to right, iterative:

```
I -> i / i k* ____
I -> u / u k* ____
```

(2) In this question, we will use finite state transducers (FSTs) for word segmentation in Chinese.
The finite state machine below is an extremely simple grammar for sentences in Chinese using this lexicon. However, this FSM does not generate spaces between the words. For example, this FSM generates the string: *wo3wang4bu4liao3jie3fang4jie1zai4na3li3*.



The lexicon used by this FSM is given in the table below.

Chinese word (pinyin)	English translation
da4	big
da4jie1	avenue
wo3	I
fang4	place
jie1	avenue
jie3fang4	liberation
bu4	not
liao3jie3	understand
wang4	forget
wang4bu4liao3	unable to forget
fang4da4	enlarge
na3li3	where
zai4	at

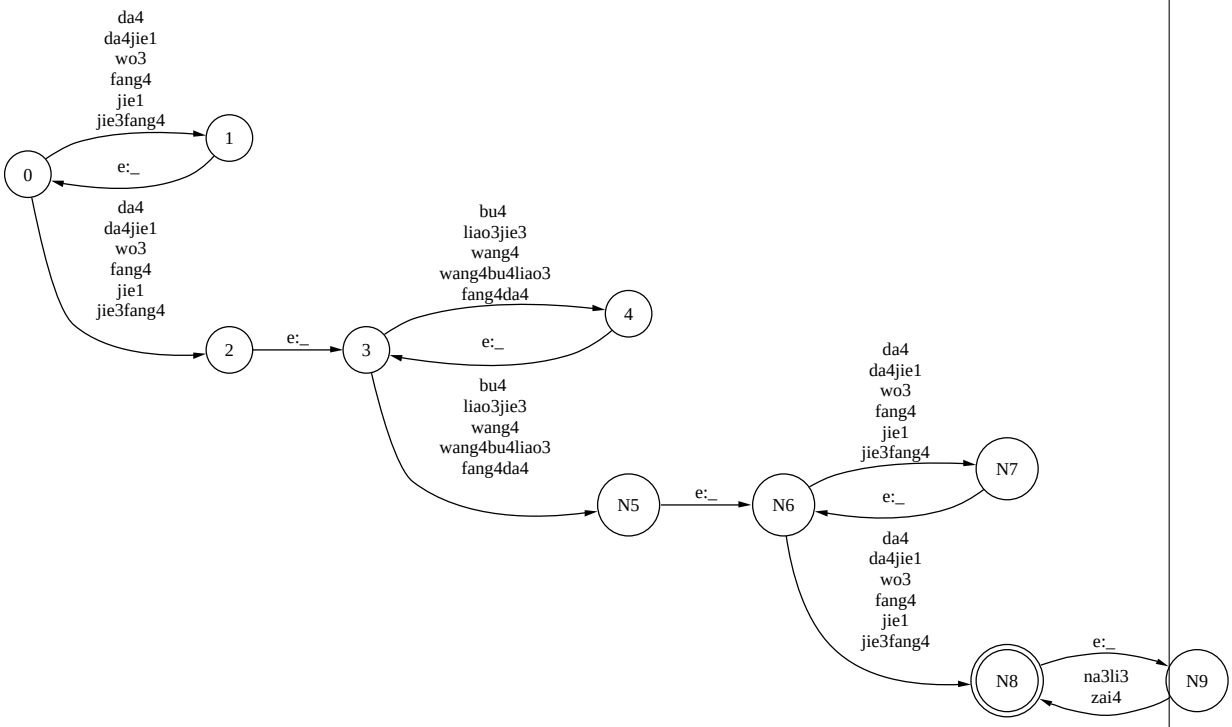
Table 1: Lexicon for Question 2.

a. Construct an FST T_{2a} where the input is a Chinese sentence with no spaces between words as accepted by the FSM above and the output has a space between any two words defined in Table 1, thus segmenting the input sentence into words. Denote the space character as ␣ in your FST.

You can use the following symbols to save time when drawing your FST:

Symbol	Value
A	`da4 da4jie1 wo3 fang4 jie1 jie3fang4`
B	`bu4 liao3jie3 wang4 wang4bu4liao3 fang4da4`
C	`na3li3 zai4`

Answer:



b. Use the following Chinese sentence without spaces between words as an input to T_{2a} and produce at least two different word segmentations (that is: two sentences with spaces between words corresponding to different word segmentations):
wo3wang4bu4liao3jie3fang4jie1zai4na3li3

Answer:

- *wo3 wang4 bu4 liao3jie3 fang4 jie1 zai4 na3li3*
- *wo3 wang4bu4liao3 jie3fang4 jie1 zai4 na3li3*

c. Using the English entries in Table 1, provide one word segmentation of the Chinese sentence from Question 2b that is closest in meaning to the following English sentence:
I was unable to forget where Liberation Avenue is.

Answer:

- *wo3(I) wang4bu4liao3(unable to forget) jie3fang4(Liberation) jie1(Avenue) zai4(at) na3li3(where)*

(3) **Edit distance**: Assume insertion of a character has cost 1, deletion has cost 1, and substitution of one character for another has cost 2.

a. What is the minimum edit distance value between target word *goal* and source word *hole*?

Answer:
`4pts:`

`levenshtein distance = 4`

b. The following is a visual display of one possible alignment between target word *goal* and source word *hole*. Using the same visual display notation, provide all other alignments that have the same edit distance.

```
g o a l _
  |   |
h o _ l e
```

Answer:

```
6pts:

levenshtein distance = 4
alignment number 1 for [4,4]:
_ g o a l _
    |   |
h _ o _ l e

alignment number 2 for [4,4]:
g _ o a l _
    |   |
_ h o _ l e

alignment number 3 for [4,4]:
g o a l _
  |   |
h o _ l e

total of 3 alignments
```

(4) **Language Modeling**

Consider a language model over character sequences that computes the probability of a word based on the characters in that word, so if word $w = c_0, c_1, \ldots, c_n$ then $P(w) = P(c_0, \ldots, c_n)$. Let us assume that the language model is defined as a bigram character model $P(c_i \mid c_{i-1})$ where

$$P(c_0, \ldots, c_n) = \prod_{i=1,2,\ldots,n} P(c_i \mid c_{i-1}) \tag{1}$$

For convenience we assume that we have explicit word boundaries: $c_0 = \texttt{bos}$ and $c_n = \texttt{eos}$ where `bos` stands for *begin sentence marker* and `eos` stands for *end of sentence marker*.

Based on this model, for the English word *booking* the probability would be computed as:

$$P(booking) = P(b \mid \texttt{bos}) \times P(o \mid b) \times P(o \mid o) \times P(k \mid o) \times P(i \mid k) \times P(n \mid i) \times P(g \mid n) \times P(\texttt{eos} \mid g)$$

The inflection *ing* is a suffix and is generated after the stem *book* with probability

$$P(ing) = P(i \mid k) \times P(n \mid i) \times P(g \mid n) \times P(\texttt{eos} \mid g)$$

In Semitic languages, like Arabic and Hebrew, the process of inflection works a bit differently. In Arabic, for a word like *kitab* the stem would be *k-t-b* where the place-holders '-' for inflection characters have been added for convenience. We will assume that each word is made up of a sequence of consonant-vowel sequences CVCVCV… and the vowels always form the inflection.

a. Provide the definition of an n-gram model that will compute the probability for the word *kitab* and *k-t-b* as follows:

$$P(kitab) = P(k \mid \texttt{bos}) \times P(t \mid k) \times P(b \mid t) \times P(i \mid b) \times P(a \mid i) \times P(\texttt{eos} \mid a)$$

$$P(k\text{-}t\text{-}b) = P(k \mid \texttt{bos}) \times P(t \mid k) \times P(b \mid t) \times P(\text{-} \mid b) \times P(\text{-} \mid \text{-}) \times P(\texttt{eos} \mid \text{-})$$

Write down the equation for this n-gram model in the same mathematical notation as equation (1).

Answer:

$$P(c_0, \ldots, c_n) = \begin{cases} \prod_{i=1}^{n} P(c_i \mid c_{i-1}) & \text{if } n \leq 3 \\ \left(P(c_1 \mid c_0) \times \prod_{i=3,5,\ldots}^{\ell} P(c_i \mid c_{i-2})\right) \times & \\ \left(P(c_2 \mid c_{\ell_o}) \times \prod_{i=4,6,\ldots}^{\ell} P(c_i \mid c_{i-2}) \times P(c_n \mid c_{\ell_e})\right) & \text{if } n > 3 \end{cases}$$

Define $\ell = n - (n \mod 2)$ and ℓ_o is the last odd number less than ℓ and ℓ_e is the last even number less than ℓ. As long as the boundary cases are right for the bigrams, we don't penalize off by one in the length, and we don't penalize for $n \leq 3$.

b. Using your n-gram model show how $P(kitab) = P(ktb) \times P(ia)$.

Answer:

$$\begin{aligned} P(kitab) &= P(c_0 = \texttt{bos}, c_1 = k, c_2 = i, c_3 = t, c_4 = a, c_5 = b, c_6 = \texttt{eos}) \\ &= P(ktb) \times P(ia, \texttt{eos}) \\ P(ktb) &= P(c_1 = k \mid c_0 = \texttt{bos}) \times P(c_3 = t \mid c_1 = k) \times P(c_5 = b \mid c_3 = t) \\ &\quad \text{this term corresponds to the first bracket in the eqn above} \\ P(ia) &= P(c_2 = i \mid c_{\ell_o} = c_5 = b) \times P(c_4 = a \mid c_2 = i) \times P(c_n = c_6 = \texttt{eos} \mid c_{\ell_e} = c_4 = a) \\ &\quad \text{corresponds to the second bracket in the eqn above} \end{aligned}$$

c. For bigram probabilities $P(c_i \mid c_{i-1})$, Katz backoff smoothing is defined as follows:

$$P_{katz}(c_i \mid c_{i-1}) = \begin{cases} \frac{r^*(c_{i-1}, c_i)}{r(c_{i-1})} & \text{if } r(c_{i-1}, c_i) > 0 \\ \alpha(c_{i-1}) P_{katz}(c_i) & \text{otherwise} \end{cases}$$

where $r(\cdot)$ provides the (unsmoothed) frequency from training data and $r^*(\cdot)$ is the Good-Turing estimate of the frequency r defined as follows:

$$r^*(c_{i-1}, c_i) = (r(c_{i-1}, c_i) + 1) \times \frac{n_{r(c_{i-1}, c_i)+1}}{n_{r(c_{i-1}, c_i)}}$$

where $n_{r(c_{i-1}, c_i)}$ is the number of different c_{i-1}, c_i types observed with count $r(c_{i-1}, c_i)$. We assume that linear interpolation has provided all missing $n_{r(\cdot)}$ values required.

$\alpha(c_{i-1})$ is chosen to make sure that $P_{katz}(c_i \mid c_{i-1})$ is a proper probability. Provide the equation for

$\alpha(c_{i-1})$.

Answer:

Step by step derivation below. We are just looking for the end result.

$$\begin{aligned}
\sum_{c_i}\left(\frac{r^*(c_{i-1},c_i)}{r(c_{i-1})}+\alpha(c_{i-1})P_{katz}(c_i)\right) &= 1\\
\sum_{c_i:r(c_{i-1},c_i)>0}\frac{r^*(c_{i-1},c_i)}{r(c_{i-1})}+\alpha(c_{i-1})\sum_{c_i:r(c_{i-1},c_i)=0}P_{katz}(c_i) &= 1\\
\alpha(c_{i-1})\sum_{c_i:r(c_{i-1},c_i)=0}P_{katz}(c_i) &= 1-\left(\sum_{c_i:r(c_{i-1},c_i)>0}\frac{r^*(c_{i-1},c_i)}{r(c_{i-1})}\right)\\
\alpha(c_{i-1}) &= \frac{1-\left(\sum_{c_i:r(c_{i-1},c_i)>0}\frac{r^*(c_{i-1},c_i)}{r(c_{i-1})}\right)}{\sum_{c_i:r(c_{i-1},c_i)=0}P_{katz}(c_i)}\\
\alpha(c_{i-1}) &= \frac{1-\left(\sum_{c_i:r(c_{i-1},c_i)>0}\frac{r^*(c_{i-1},c_i)}{r(c_{i-1})}\right)}{1-\left(\sum_{c_i:r(c_{i-1},c_i)>0}P_{katz}(c_i)\right)}
\end{aligned}$$

Also acceptable is the somewhat less precise answer which assumes $\sum_{c_i} P_{katz}(c_i) = 1$:

$$\alpha(c_{i-1}) = 1-\sum_{c_i}\frac{r^*(c_{i-1},c_i)}{r(c_{i-1})}$$