# 1.

## a)

### *i*)

since α is actually soft-max function with each value somewhat between 0 to 1. Summing up all of them result in 1. And this means soft-max function is probability distribution. And the categorical case is we have *n* number of elements with their own probability (0 *< pi < 1).*

### *ii*)

*query* for a token gets dot product with all of other tokens *key*. This value receives high amount whenever this dot product gets high (*kjT.q >> kiTq).* This case happens when all *key* values of *j* tokens are large numbers and they’re same sign with q element-wise. And in this case we have much weight on j token.

### *iii*)

The output of previous question is when α put much weight on specific token called *j*. in this case *c* will get higher attention on *j* token and approximately focus nothing to other tokens. This will cause *c* to get almost *vj value.*

### *iv)*

for example, if we have sentence which is tokenized to [“I”, “sat”, “by”, “the”, “river”, “side”] and the dot production of “side” *query(q)* with “river” *key* is very high but with other keys are very low, it means that “side” token just focus on the “river” token and gets “river” value(copied) but other values aren’t considered much.

## b)

### *i)*

as it was mentioned:

*va = c1a1 + c2a2 + … + cmam 🡪 in which c is just constant vector where each element has constant value*

*should show that:*

*Ms = va 🡪 M(va + vb) = va 🡪 Mva + Mvb = va 🡪 so two deduction should be made :*

1. *Mvb = 0*
2. *Mva  = va*

*So we should consider M* *in some way that Matrix Product with vb get 0. And Matrix Product with va act like that its eye(I) matrix.*

*a1T*

*…*

*amT*

*If we consider M like this: M = = AT*

*Mva = [a1T a1 c1 , a2Ta2 c2, ……, amT am cm] 🡪 we know from the fact that a’s are orthogonal and normalized so aiT.ajT = 0 , aiT. aiT = 1*

*Mvb = [a1T b1 x1 , a2Tb2 x2, ……, amT bm xm] 🡪 we also know that a and b are orthogonal so the whole matrix get 0.*

*Mva + Mvb = [c1, c2, …, cm] + [0, 0, …, 0] = [c1, c2, …, cm] 🡪 this matrix should be equal to va and we know that va is some constant element of c.*

*So we deduce that M = AT.*

### *ii)*

we want at the end this equation : ***c ≈ ½ (va + vb)***

*it means that values of other keys should be ignored and just consider a and b key.*

*So for winning of va and vb we should make their* α *very large to other keys so then we should have*

*Ka q ≈ kb q 🡪 and this value should be very much large*

*Ki q ≈ 0 🡪 for every i not equal to a,b.*

*So we should pick q in such way that it contains both ka and kb so that the dot product gets large.*

*We can consider q = L(ka + kb) 🡪 L is some large number*

*And as the question said all keys are orthogonal and normalized to 1 so for every i not equal to a,b we have ki . q = ki . L(ka + kb) = 0 🡪 because i is orthogonal to a, b*

*And ka q = kb q = ka . L(ka + kb) = L 🡪 because a to a is normalized and then get 1 at matrix product*

*So when we calculate alpha:*

α*a* = αb =

## c)

### *i)*

here question is almost same as what we have proposed in the previous question. Since that *covariance* matrix is multiplied by vanishingly small α, the effect of *covariance* gets almost 0. So we can almost say that the equation *ki ≈ µi* holds true. And all µ are perpendicular we can result that we should consider *q* like previous question.

***q = L(µa + µb) 🡪 L is some large number***

### *ii)*

as in the question said we should assume Σa = αI + ½(µaµaT)

and α is vanishingly small and since µ is normalized to 1 we have Σa ≈ 0 + ½ = ½ and µ = 1

so we have*:*

*ka = εµa , ε ~ N(1, 0.5)*

*and k for other i is just same as before equal µi.*

*three cases :*

* *i ≠ a,b 🡪 q.k = 0*
* *i = a 🡪 q.k = εL*
* *i = b 🡪 q.k = L*

*finally we have c = αa.va +* αb.*vb and :*

c *=*

*when ε is minimum i.e 0.5 🡪 c ≈ vb*

*when ε is maximum i.e 1.5 🡪 c ≈ va*

## d)

### *i)*

there can be multiple solution for this question:

* ½ (c1 + c2) = ½ (va + vb) 🡪 c1 + c2 = va + vb

We can design in a way that c1 = va , c2 = vb

And this can be achieved by **q1 = L.µa** , where L is large number and same for **q2 = L. µb**

Because µ are mutually orthogonal so for c1 we just have value of q1 which is just copy of µa and same happens for q2

* Or we can just use combination of them just like previous question

q1 = q2 = L(ka + kb) 🡪 αa = αb = ½ 🡪 c1 = ½ (va + vb) , c2 = ½ (va + vb)

½(c1+c2) = ½(va + vb)

### *ii)*

as in the question *c part ii* we discussed there are the fact that we have

we should assume Σa = αI + ½(µaµaT)

and α is vanishingly small and since µ is normalized to 1 we have Σa ≈ 0 + ½ = ½ and µ = 1

so we have*:*

*ka = εµa , ε ~ N(1, 0.5)*

*and k for other i is just same as before equal µi.*

*and what we have designed for query vector in the previous question was*

q1 = L.µa , q2 = L. µb then :

q1.ka = q1. *ε*µa = L.µa. *ε*µa = *L.ε 🡪 α1 ≈ exp(L. ε) / exp(L. ε) ≈ 1*

q2.kb = q2.µb = L.µb.µb = *L 🡪 α2 ≈ exp(L) / exp(L) ≈ 1*

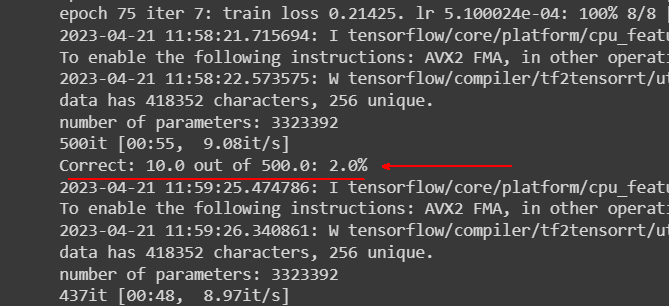
*so we can see that c1 = va and c2 = vb*

and in multi head attention we should take average : c = ½(c1 + c2) = ½(va + vb)

# 2.

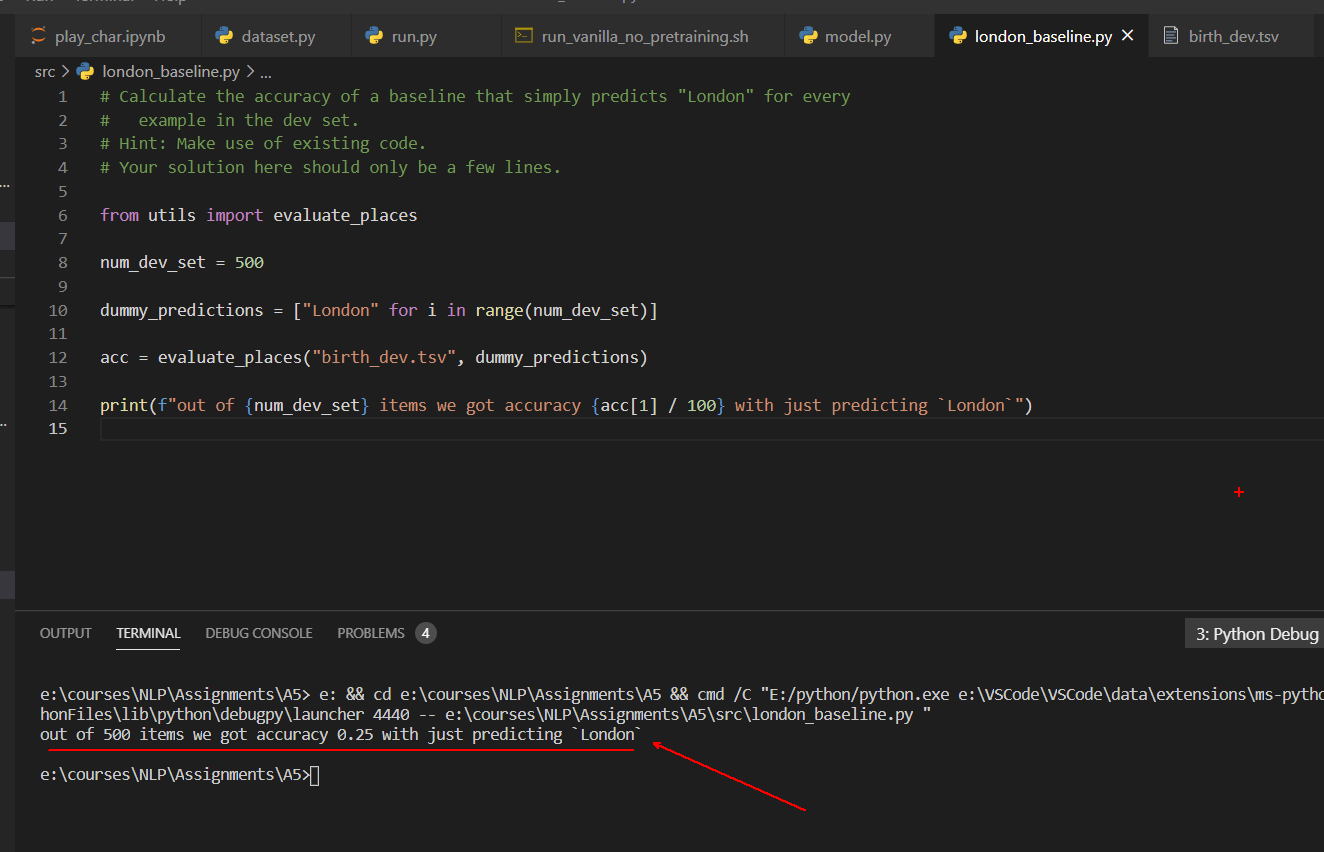
## d)

after complete *TODO* code I received accuracy of :



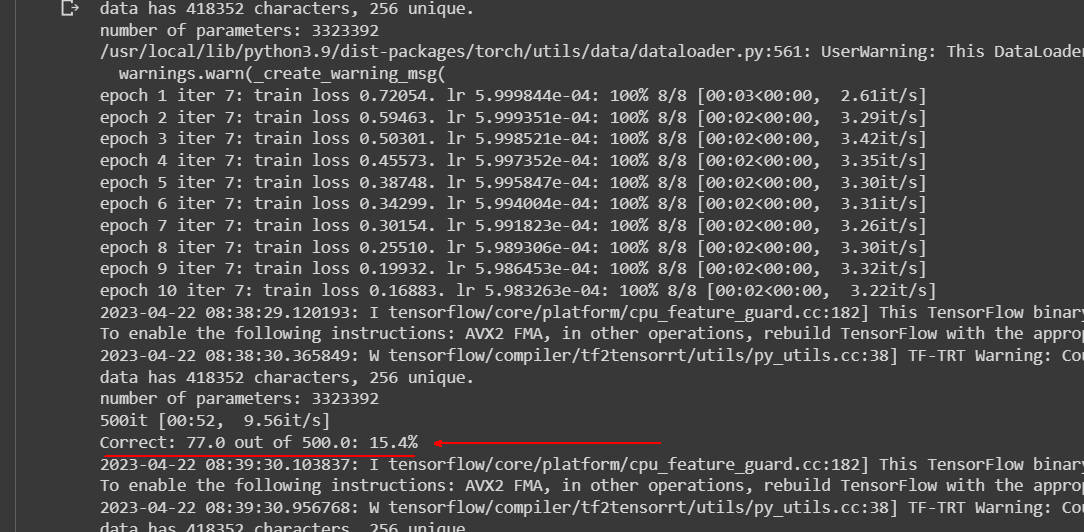
It’s 2% which is highly low.

So I just make dummy prediction with London for each evaluation dataset I got following result :



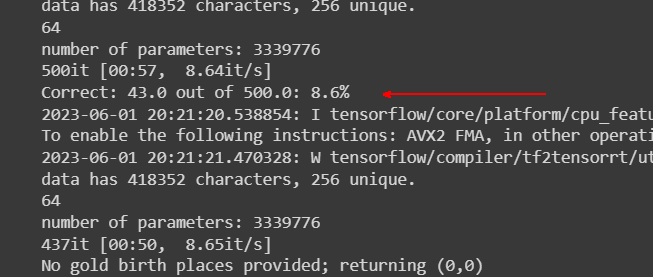
## f)

after doing pretraining on *wiki.txt* we have achieved following result:



## g)

### i)



### ii)

first let’s see computational cost of main transformer(vanilla):

* Self-attention: *ℓ2.d 🡪*  *ℓ is : sequence length , d is : number of dimension*
* Attention head: *ℓ2.d.h 🡪 h is : number of heads*
* Feed forward: *ℓ.d.f 🡪 f is : hidden size*
* Number of transformer layer: *L(ℓ2.d.h + ℓ.d.f)*

Now let’s compare it with Perceiver model:   
first and last layer of this model use down Projection and up Projection respectively.  
so L becomes (*L – 2). (m2.d.h + m.d.f) + down projection complexity + up projection complexity(m is bottleneck\_dim).*

Notice that *ℓ* becomes *m* because we’re doing our computation on lower dimensionality.

*down projection complexity =*  *ℓ.m.d.h + m.d.f*

*up projection complexity = ℓ.m.d.h + ℓ.d.f*

Perceiver complexity : (*L – 2). (m2.d.h + m.d.f) + 2.ℓ.m.d.h + d.f(ℓ + m)*

# 3.

## a.

pretraining is important phase for our model to become more general and see more common data and casual text on open domain. As we see we trained our model in pretraining phase on 600 epochs to learn whatever it must learn between text correlation and understand the concept. And we can customize our model by fine tuning on our specific task which leads in a far better result.

## b.

**Information misleading in important situation :** our models keep generating answers that we don’t know even its correct or not and since it doesn’t provide any explanation that how it produced that we have no idea and consider that as correct birth place.

**Example :** imagine a lecturer who is lecturing about famous people and taking their birth place data from our app when he lectures with some wrong places people will doubt about his/her knowledge and don’t pay attention to him.

**Losing Trust:** if the system frequently presents users with incorrect or unreliable birth places, it can erode trust in the system's overall accuracy and credibility. Users may become suspicious of the system's output and lose confidence in its ability to provide accurate information.

**Example:** consider an voice assistant which uses our model app api for birth places. When people start to ask birth places of people our voice assistant uses api and then generate fake birth place. And people start to lose trust in our voice assistant and this may lead to bankruptcy of our investment on our voice assistant.

## c.

one possible way is that our model analyzes test example and see its embedding tokenization and then map our test example question to the question in training set sample which is most similar and then use that output for test example. Or it may produce some response which is just following general patterns seen at training time or fine-tuning and has some meaning but the answer isn’t close to the real answer. And as the last possibility if the test provided isn’t close even a little to our training data it may generate some random and non-sense answers without any meaning.

**concerns :** the possibilities that has been mentioned may cause our app to lose member because making non-sense answers without any explanations. And can be very embarrassing when we’re using our app for official paper and this may cause bad reputation for our company.