

# 이게 왜 필요할까

- feature extraction 이후 유사도 비교를 하는 태스크가 많음
- 해당 임베딩을 서빙 코드에서 관리하는 경우 파편화 문제 존재
- CPU, GPU, NPU에서 코드 변경없이 모델과 임베딩을 관리하고 성능까지 높이고 싶다.
- 배포가 편해지는 것은 덤으로 얻고 싶다.

# 하려는 것, 예상되는 효과

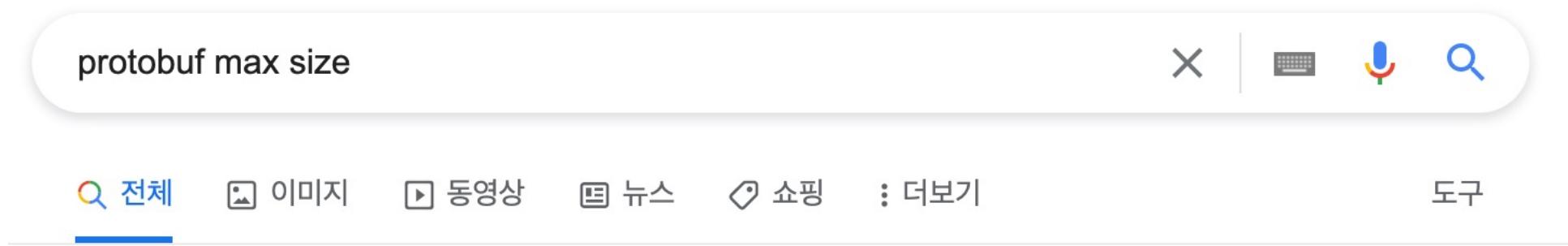
- stored embeddings 된 임베딩과 feature extraction 모델의 계산 그래프를 결합한다.
- CPU 서빙 관점에서는 계산 속도는 그대로 모델 로딩 속도는 감소 코드의 배포 용이성이 증가
- GPU 관점에서는 CPU의 장점 + 메모리 복사 대폭 감소로 인하여 현저한 속도 개선
- 덤으로 AWS M1 인스턴스의 뉴럴 코어나 TPU, NPU 종류도 사용 가능 사용 가능

# 사내 세미나 용도로 만들었는데

- 사내에서 세션으로 하기엔 지나치게 specific 한 문제
- Tensorflow 모델 저장이 2GB 제한 있음
- 구글 protocol buffer 자체 스펙 문제
- 5년간 해결 안됨
- 텐서 내부 OP 들이나 variable 은 lazy 한 모델 구조로 가능
- 그러나... 외부 변수나 임베딩들이 안됨! 배포할 때 골치 아픔
- 해결해보자
- <https://github.com/tensorflow/tensorflow/issues/51870#>

# Protobuf has a hard limit of 2GB

- 물리적으로  $2^{31} - 1$  까지가 크기 범위



## 2GB

Protobuf has a hard limit of **2GB**, because many implementations use 32-bit signed arithmetic. For security reasons, many implementations (especially the Google-provided ones) impose a size limit of 64MB by default, although you can increase this limit manually if you need to. 2015. 12. 7.

<https://stackoverflow.com/questions/google-protobuf-...>

# 물론 잘 안됩니다.

```
/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/ops/var_tables.py:67 getter
    return captured_getter(captured_previous, **kwargs)
/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/eager/def_function.py:750 variable_capturing_scope
    v = UnliftedInitializerVariable(
/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/ops/variables.py:264 __call__
    return super(VariableMetaclass, cls).__call__(*args, **kwargs)
/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/eager/def_function.py:299 __init__
    initial_value = ops.convert_to_tensor(initial_value,
/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/profiler/trace.py:163 wrapped
    return func(*args, **kwargs)
/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/framework/ops.py:1566 convert_to_tensor
    ret = conversion_func(value, dtype=dtype, name=name, as_ref=as_ref)
/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/framework/tensor_conversion_registry.py:52 _default_conversion
    return constant_op.constant(value, dtype, name=name)
/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/framework/constant_op.py:264 constant
    return _constant_impl(value, dtype, shape, name, verify_shape=False,
/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/framework/constant_op.py:281 _constant_impl
    tensor_util.make_tensor_proto(
/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/framework/tensor_util.py:527 make_tensor_proto
    raise ValueError()
```

ValueError: Cannot create a tensor proto whose content is larger than 2GB.

# Tf.function 의 jit를 이용하여

- <https://runebook.dev/ko/docs/tensorflow/function>
- 콘크리트 함수 이용해서 lazy한 로딩으로 저장

```
class MyModule(tf.Module):
    def __init__(self):
        self.v = None

    @tf.function
    def __call__(self, x, k):
        #with tf.device('/CPU:0'):
        if True:
            if self.v is None:
                # 4 GiB variable
                self.v = tf.Variable(tf.random.uniform([20000000, 100], dtype=tf.dtypes.float32))
            cos = keras.losses.CosineSimilarity(axis=1, reduction=tf.keras.losses.Reduction.NONE)
            candidate = cos(x, self.v) * -1
            topk = tf.math.top_k(candidate, k)
            return topk

func = MyModule()

cfunc = func.__call__.get_concrete_function(tf.TensorSpec(None, tf.float32), tf.TensorSpec(None, tf.int32))
tf.saved_model.save(func, 'my_saved_model111', signatures=cfunc)
```

# 되긴하지만 다른 문제가 있음

- <https://github.com/tensorflow/tensorflow/issues/51870#>

```
$ tree my_saved_model/
my_saved_model/
├── assets
└── saved_model.pb
└── variables
    ├── variables.data-00000-of-00001
    └── variables.index

2 directories, 3 files

$ ls -lh my_saved_model/variables
total 4.1G
-rw-r--r-- 1 root root 4.1G Sep  8 23:58 variables.data-00000-of-00001      # <===== Larger than 2GBs
-rw-r--r-- 1 root root  211 Sep  8 23:58 variables.index
```

# 알려진 모든 방법으로 안된다는 걸 알게 됨

```
File "/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/eager/def_function.py", line 1367, in get_concrete_function
    concrete = self._get_concrete_function_garbage_collected(*args, **kwargs)
File "/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/eager/def_function.py", line 1273, in _get_concrete_function_garbage_collected
    self._initialize(args, kwargs, add_initializers_to=initializers)
File "/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/eager/def_function.py", line 763, in _initialize
    self._stateful_fn._get_concrete_function_internal_garbage_collected( # pylint: disable=protected-access
File "/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/eager/function.py", line 3050, in _get_concrete_function_internal_garbage_collected
    graph_function, _ = self._maybe_define_function(args, kwargs)
File "/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/eager/function.py", line 3444, in _maybe_define_function
    graph_function = self._create_graph_function(args, kwargs)
File "/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/eager/function.py", line 3278, in _create_graph_function
    graph_function = ConcreteFunction(
File "/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/eager/function.py", line 1597, in __init__
--self._delayed_rewrite_functions = _DelayedRewriteGradientFunctions(
File "/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/eager/function.py", line 676, in __init__
--self._inference_function = _EagerDefinedFunction(
File "/home/noah/anaconda3/lib/python3.9/site-packages/tensorflow/python/eager/function.py", line 494, in __init__
--function_def.ParseFromString(compat.as_bytes(proto_data))
google.protobuf.message.DecodeError: Error parsing message with type 'tensorflow.FunctionDef'
(base) noah@noah-go:~$
```

# 지난 5년간 해결된 사례가 없는 문제

- <https://github.com/tensorflow/tensorflow/issues/51870#>
- <https://stackoverflow.com/questions/51244489/valueerror-cannot-create-a-tensor-proto-whose-content-is-larger-than-2gb>
- <https://stackoverflow.com/questions/35394103/initializing-tensorflow-variable-with-an-array-larger-than-2gb>
- <http://ostack.cn/?qa=604286/>

# 지난 5년간 해결된 사례가 없는 문제

- Tensorflow에 PR 보내면 좋을 것 같은데..
- 토스팀으로 보낼까요? 함께 하실 분 구합니다.
- Tensorflow는 범용 연산 그래프를 생성하는 도구로 거의 방향성을 가는 중,  
언어나 런타임 플랫폼이나 비지니스 의존성이 적음 거의 JAVA와 같은 형태  
-> 2GB 변수 제한은 치명적
- <https://www.tensorflow.org/guide/function?hl=ko>
- <https://www.tensorflow.org/xla?hl=ko>

# 답이 없다고 하니 역공학으로 접근해봅니다

```
import numpy as np  
  
data = np.random.random((50000000, 100))  
np.save(data, "0.npy")
```

```
(base) noah@noah-go:~/data$ ls -all  
total 80078420  
drwxrwxr-x 2 noah noah 4096 12월 18 01:30 .  
drwxr-xr-x 38 noah noah 4096 12월 20 02:18 ..  
-rw-rw-r-- 1 noah noah 2000000128 12월 17 13:48 0.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:53 10.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:53 11.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:53 12.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:53 13.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:53 14.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:53 15.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:53 16.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:53 17.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:53 18.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:53 19.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:52 1.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:53 20.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:53 21.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:54 22.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:54 23.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:54 24.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:54 25.npy  
-rw-rw-r-- 1 noah noah 2000000128 12월 5 01:54 26.npy
```

# 답이 없다고 하니 역공학으로 접근해봅니다

- 저장된 용량이 2억 128 바이트로 나뉘는 걸 보니  
데이터 부 2억 바이트 헤더 부 128 바이트임 알 수 있습니다.

00000000	93 4E 55 4D	50 59 01 00	76 00 7B 27	64 65 73 63	72 27 3A 20	27 3C 66 34	.NUMPY..v.{'descr': '<f4
00000018	27 2C 20 27	66 6F 72 74	72 61 6E 5F	6F 72 64 65	72 27 3A 20	46 61 6C 73	', 'fortran_order': Fals
00000030	65 2C 20 27	73 68 61 70	65 27 3A 20	28 35 30 30	30 30 30 30	2C 20 31 30	e, 'shape': (5000000, 10
00000048	30 29 2C 20	7D 20 20 20	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 20	0), }
00000060	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 20	20 20 20 20	
00000078	20 20 20 20	20 20 20 0A	8B A5 72 3D	23 33 60 3E	69 FC FC 3E	91 8E 5F 3F	...r=#3`>i...>..._?
00000090	8B 00 21 3F	1E CC 36 3E	86 5C 6B 3E	04 85 59 3F	12 70 B4 3C	94 5D 1B 3F	..!?.6>.\k>..Y?.p.<.].?
000000A8	86 2C 64 3F	2A 75 F4 3E	E6 81 61 3F	65 A7 5C 3F	03 96 40 3F	25 59 97 3E	,d?*u.>..a?e.\?..@%Y.>
000000C0	18 EA 68 3E	D5 30 61 3F	9E 90 0E 3D	13 3F 83 3E	B1 F3 6D 3F	94 A1 21 3F	..h>.0a?...=.=?..>..m?..!?
000000D8	3D 54 BC 3E	59 94 B2 3E	ED E4 DC 3E	BF 5B 70 3F	98 0D 1B 3F	83 B7 51 3E	=T.>Y..>....>.[p?...?..Q>
000000F0	5F 30 2F 3F	25 B5 3B 3F	72 82 EC 3E	18 D7 3C 3E	46 EF 24 3F	80 04 AF 3D	_0/?%.;?r..>..<F.\$?...=
00000108	1D E3 56 3F	45 27 1A 3F	49 FF 23 3F	7C 34 1F 3F	8D 78 89 3E	63 41 5D 3F	..V?E'.?I.#?14.?..x.>cA]?
00000120	C2 D1 90 3D	06 EC 33 3E	18 9B C7 3E	CB EB CA 3D	34 D2 99 3E	3E 51 32 3F	...=..3>....>....=4..>>Q2?
00000138	54 C5 AB 3E	23 0A 78 3F	AA 19 75 3F	DE 7E 18 3E	0A 98 9B 3E	65 16 14 3F	T..>#.x?..u?..~.>....>e...?

# Lazy 모델 생성으로 분리한 변수부

```
(base) noah@noah-go:~/my_saved_model/variables$ ls -all
total 15625028
drwxr-xr-x 2 noah noah      4096 12월 20 01:30 .
drwxr-xr-x 5 noah noah      4096 12월 20 01:30 ..
-rw-rw-r-- 1 noah noah 8000000092 12월 19 22:47 variables.data-00000-of-00001
-rw-rw-r-- 1 noah noah 8000000092 12월 20 01:30 variables.data-00000-of-00001_old
-rw-rw-r-- 1 noah noah      207 12월 20 01:04 variables.index
(base) noah@noah-go:~/my_saved_model/variables$
```

DCD64F20	82 D3 4D 3F	CD 71 DD 3E	79 D8 00 3F	62 82 8F 3E	E3 2F 18 3E	B8 1A CF 3E	..M?.q.>y..?b..>./.>...>
DCD64F38	4F 4F 84 3C	B6 4C AA 3E	F9 1B C1 3E	A1 11 14 3F	12 E2 8B 3D	E2 D5 74 3F	00.<.L.>...>...?...=..t?
DCD64F50	39 93 65 3F	6E 51 78 3C	2A 2C F5 3D	06 8D 7F 3F	A9 43 DF 3E	01 35 F7 3E	9.e?nQx<*,.=...?.C.>.5.>
DCD64F68	0C 74 57 3F	FC B4 34 3F	1A C7 85 3E	89 1E C1 3E	77 30 AA 3E	82 4E 65 3F	.tW?..4?...>...>w0.>.Ne?
DCD64F80	CC 8F D8 3E	5F 8E A5 3E	47 C8 7F 3F	9D 12 3B 3F	29 5B 07 3E	F8 73 26 3F	...>_...>G..?..;?)[.>.s&?
DCD64F98	6B 8E 38 3F	E2 2D A4 3E	9D 04 F1 3D	13 DA 1E 3F	17 94 48 3F	A7 61 0E 3F	k.8?.-.>...=...?..H?.a.?
DCD64FB0	F2 76 8B 3E	1C 29 2A 3F	D1 06 5B 3F	8F EB 80 3E	06 53 A1 3E	43 8A C5 3C	.v.>.)*?...[?...>.S.>C..<
DCD64FC8	05 8F 13 3F	70 D7 79 3D	0E 9B 0C 3F	6F A9 99 3E	FB 22 F6 3C	7D 60 71 3F	...?p.y=...?o..>.".<}`q?
DCD64FE0	C6 FE 5F 3F	3F 84 C3 3D	F8 8B 42 3F	BB 3F 21 3F	39 52 AF 3E	D9 C6 AE 3E	...??..=.B?.?!.?9R.>...>
DCD64FF8	36 49 7E 3F	54 74 03 3F	57 22 12 31	DE 0A 17 0A	05 08 01 12	01 76 0A 0E	6I~?Tt.?W".1.....v..
DCD65010	08 02 12 0A	73 69 67 6E	61 74 75 72	65 73 0A 3A	12 38 0A 0E	56 41 52 49	....signatures.:..8..VARI
DCD65028	41 42 4C 45	5F 56 41 4C	55 45 12 08	56 61 72 69	61 62 6C 65	1A 1C 76 2F	ABLE_VALUE..Variable..v/
DCD65040	2E 41 54 54	52 49 42 55	54 45 53 2F	56 41 52 49	41 42 4C 45	5F 56 41 4C	.ATTRIBUTES/VARIABLE_VAL
DCD65058	55 45 0A 00						UE..
--- variables.data-00000-of-00001 --0x1DCD65000/0x1DCD6505C-----							



# 그래서 데이터 바이너리 본체를 치환해 봄

- 무언가 체크섬 변조로직에 걸렸습니다.

```
at akka.actor.ActorCell.invoke(ActorCell.scala:548)
at akka.dispatch.Mailbox.processMailbox(Mailbox.scala:270)
at akka.dispatch.Mailbox.run(Mailbox.scala:231)
at akka.dispatch.Mailbox.exec(Mailbox.scala:243)
at java.base/java.util.concurrent.ForkJoinTask.doExec(ForkJoinTask.java:290)
at java.base/java.util.concurrent.ForkJoinPool$WorkQueue.topLevelExec(ForkJoinPool.java:1020)
at java.base/java.util.concurrent.ForkJoinPool.scan(ForkJoinPool.java:1656)
at java.base/java.util.concurrent.ForkJoinPool.runWorker(ForkJoinPool.java:1594)
at java.base/java.util.concurrent.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:183)
Caused by: org.tensorflow.exceptions.TensorFlowException: 2 root error(s) found.
(0) DATA_LOSS: TensorBundle at /home/noah/my_saved_model/variables/variables shard 0 (8000000000 bytes): Checksum does not match: stored 4105128064 vs. calculated on the restored bytes 96286534
[[{{node RestoreV2}}]]
[[StatefulPartitionedCall_2/RestoreV2/_9]]
(1) DATA_LOSS: TensorBundle at /home/noah/my_saved_model/variables/variables shard 0 (8000000000 bytes): Checksum does not match: stored 4105128064 vs. calculated on the restored bytes 96286534
[[{{node RestoreV2}}]]
0 successful operations.
0 derived errors ignored.
    at org.tensorflow.internal.c_api.AbstractTF_Status.throwExceptionIfNotOK(AbstractTF_Status.java:101)
    at org.tensorflow.SavedModelBundle.load(SavedModelBundle.java:623)
    at org.tensorflow.SavedModelBundle.access$000(SavedModelBundle.java:67)
    at org.tensorflow.SavedModelBundle$Loader.load(SavedModelBundle.java:97)
    at org.tensorflow.SavedModelBundle.load(SavedModelBundle.java:357)
    at similarity.tensor.PbLoader$.apply(PbLoader.scala:20)
    at similarity.service.VectorSimilarityService$.<init>(VectorSimilarityService.scala:24)
    at similarity.service.VectorSimilarityService$.<clinit>(VectorSimilarityService.scala)
      ... 56 more
```

# 데이터만 치환한 두개의 메타 파일을 diff

- 데이터의 타입, 길이가 동일하다면 두개의 4바이트 변수만 다름

n vs r	
0	00000608 011A0208 01001C11 5F434845 434B504F 494E5441 424C455F 4F424A45
32	43545F47 52415048 08071200 2080A8D6 B907285C 3547FC92 52001C1A 762F2E41
64	54545249 42555445 532F5641 52494142 4C455F56 414C5545 0801120B 120508C0
96	96B10212 02086428 80A8D6B9 07356FA1 86D80000 00000100 000000D0 4C492A00
128	00000001 00000000 C0F2A1B0 00010277 007A0000 00000100 000000FE E3DA747F
160	088C010E 00000000 00000000 00000000 00000000 00000000 00000000 00000000
192	00000000 00000057 FB808B24 7547DB

# Tensorflow 코어 로직을 열어봄

- 하지만 아무리 찾아봐도 변경된 데이터 부분의 체크섬과 동일한 바이너리 값이 없음
- Tensorflow로 zerofill로 생성한 더미 데이터와 우리가 서빙하고 싶은 임베딩 데이터의 체크섬 각각

```
>>> crc32c.crc32c(open("variables.data-00000-of-00001","rb").read(8000000000))
96286534
>>> crc32c.crc32c(open("/home/noah/index/data","rb").read(8000000000))
410512806
```

# 마스크 사용하는 간단한 암복화로직 발견

```
#include <stdio.h>
#include <stdint.h>

static const u_int kMaskDelta = 0xa282ead8ul;
u_int Mask(u_int crc) {
    return ((crc >> 15) | (crc << 17)) + kMaskDelta;
}

u_int Unmask(u_int masked_crc) {
    uint rot = masked_crc - kMaskDelta;
    return ((rot >> 17) | (rot << 15));
}

int main(){
printf("%x", Mask(4105128064u));
printf("%x", Mask(96286534u));
}
```

# 적용해 봅니다.

New paste:

Language: C++ ▾

```
static const u_int kMaskDelta = 0xa282ead8ul;
u_int Mask(u_int crc) {
    return ((crc >> 15) | (crc << 17)) + kMaskDelta;
}

u_int Unmask(u_int masked_crc) {
    uint rot = masked_crc - kMaskDelta;
    return ((rot >> 17) | (rot << 15));
}

int main(){
printf("%x", Mask(4105128064u));
printf("%x", Mask(96286534u));
}
```

Private [?]

Run code

Submit

Output:

1	3384d436
2	110ef652

# 체크섬을 재 계산해서 파일에 넣습니다.

n vs r	
0	00000608 011A0208 01001C11 5F434845 434B504F 494E5441 424C455F 4F424A45
32	43545F47 52415048 08071200 2080A8D6 B907285C 3547FC92 52001C1A 762F2E41
64	54545249 42555445 532F5641 52494142 4C455F56 414C5545 0801120B 120508C0
96	96B10212 02086428 80A8D6B9 07356FA1 86D80000 00000100 000000D0 4C492A00
128	00000001 00000000 C0F2A1B0 00010277 007A0000 00000100 000000FE E3DA747F
160	088C010E 00000000 00000000 00000000 00000000 00000000 00000000 00000000
192	00000000 00000057 FB808B24 7547DB
00	00000608 011A0208 01001C11 5F434845 434B504F 494E5441 424C455F 4F424A45
20	43545F47 52415048 08071200 2080A8D6 B907285C 3547FC92 52001C1A 762F2E41
40	54545249 42555445 532F5641 52494142 4C455F56 414C5545 0801120B 120508C0
60	96B10212 02086428 80A8D6B9 0735E12C 62320000 00000100 000000F7 894B1D00
80	00000001 00000000 C0F2A1B0 00010277 007A0000 00000100 000000FE E3DA747F
A0	088C010E 00000000 00000000 00000000 00000000 00000000 00000000 00000000
C0	00000000 00000057 FB808B24 7547DB

# 그래도 안되네요

```
at akka.dispatch.Mailbox.processMailbox(Mailbox.scala:270)
at akka.dispatch.Mailbox.run(Mailbox.scala:231)
at akka.dispatch.Mailbox.exec(Mailbox.scala:243)
at java.base/java.util.concurrent.ForkJoinTask.doExec(ForkJoinTask.java:290)
at java.base/java.util.concurrent.ForkJoinPool$WorkQueue.topLevelExec(ForkJoinPool.java:1020)
at java.base/java.util.concurrent.ForkJoinPool.scan(ForkJoinPool.java:1656)
at java.base/java.util.concurrent.ForkJoinPool.runWorker(ForkJoinPool.java:1594)
at java.base/java.util.concurrent.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:183)
Caused by: org.tensorflow.exceptions.TensorFlowException: 2 root error(s) found.
(0) DATA_LOSS: Unable to read file (/home/noah/my_saved_model/variables/variables.index). Perhaps the file is corrupt or was produced by a newer version of TensorFlow with format changes (failed to seek to header entry): block checksum mismatch
    [[[{{node RestoreV2}}]]]
    [[StatefulPartitionedCall_2/RestoreV2/_9]]
(1) DATA_LOSS: Unable to read file (/home/noah/my_saved_model/variables/variables.index). Perhaps the file is corrupt or was produced by a newer version of TensorFlow with format changes (failed to seek to header entry): block checksum mismatch
    [[[{{node RestoreV2}}]]]
0 successful operations.
0 derived errors ignored.
    at org.tensorflow.internal.c_api.AbstractTF_Status.throwExceptionIfNotOK(AbstractTF_Status.java:101)
    at org.tensorflow.SavedModelBundle.load(SavedModelBundle.java:623)
    at org.tensorflow.SavedModelBundle.access$000(SavedModelBundle.java:67)
    at org.tensorflow.SavedModelBundle$Loader.load(SavedModelBundle.java:97)
    at org.tensorflow.SavedModelBundle.load(SavedModelBundle.java:357)
    at similarity.tensor.PbLoader$.apply(PbLoader.scala:20)
    at similarity.service.VectorSimilarityService$.<init>(VectorSimilarityService.scala:24)
    at similarity.service.VectorSimilarityService$.<clinit>(VectorSimilarityService.scala)
    ... 56 more
```

# 두 번째도 체크섬인듯 합니다.

n vs r	
0	00000608 011A0208 01001C11 5F434845 434B504F 494E5441 424C455F 4F424A45
32	43545F47 52415048 08071200 2080A8D6 B907285C 3547FC92 52001C1A 762F2E41
64	54545249 42555445 532F5641 52494142 4C455F56 414C5545 0801120B 120508C0
96	96B10212 02086428 80A8D6B9 07356FA1 86D80000 00000100 000000D0 4C492A00
128	00000001 00000000 C0F2A1B0 00010277 007A0000 00000100 000000FE E3DA747F
160	088C010E 00000000 00000000 00000000 00000000 00000000 00000000 00000000
192	00000000 00000057 FB808B24 7547DB
00	00000608 011A0208 01001C11 5F434845 434B504F 494E5441 424C455F 4F424A45
20	43545F47 52415048 08071200 2080A8D6 B907285C 3547FC92 52001C1A 762F2E41
40	54545249 42555445 532F5641 52494142 4C455F56 414C5545 0801120B 120508C0
60	96B10212 02086428 80A8D6B9 0735E12C 62320000 00000100 000000F7 894B1D00
80	00000001 00000000 C0F2A1B0 00010277 007A0000 00000100 000000FE E3DA747F
A0	088C010E 00000000 00000000 00000000 00000000 00000000 00000000 00000000
C0	00000000 00000057 FB808B24 7547DB



새 파일



파일 열기



다른 이름으로 저장



실행 취소



다시 실행



도구



번역



설정



도움말

PATREON

PayPal

## 파일 정보

## -제목 없음- ×

## variables.index ×

파일 이름

variables.index

000000000

파일 크기

207바이트

00000016

## 데이터 검사 (리틀 엔디안)

00000048

형식

부호없는  
(Unsigned) (+)부호있는 (Signed)  
(±)

00000080

8비트 정수

194

-62

00000096

16비트 정수

17346

17346

00000112

24비트 정수

13976514

-2800702

00000144

32비트 정수

3805627330

-489339966

00000160

64비트 정수 (+)

3805627330

00000176

64비트 정수 (±)

13

}

14

16비트 부동 소수점

15

int main(){

32비트 부동 소수점

16

printf("%x\n", Unmask(3805627330u));

64비트 부동 소수점

17

printf("%u", Unmask(3805627330u));

LEB128 (+)

18

}

LEB128 (±)

MS-DOS 날짜, 시간

OLE 2.0 날짜, 시간

UNIX 32비트 날짜, 시간

Macintosh HFS 날짜, 시간

Macintosh HFS+ 날짜, 시간

이진

00 00 06 08 01 1A 02 08 01 00 1C 11 5F 43 48 45

43 4B 50 4F 49 4E 54 41 42 4C 45 5F 4F 42 4A 45

43 54 5F 47 52 41 50 48 08 07 12 00 20 80 A0 D9

E6 1D 28 5C 35 47 FC 92 52 00 1C 1A 76 2F 2E 41

54 54 52 49 42 55 54 45 53 2F 56 41 52 49 41 42

4C 45 5F 56 41 4C 55 45 08 01 12 0B 12 05 08 80

DA C4 09 12 02 08 64 28 80 A0 D9 E6 1D 35 52 F6

0E 11 00 00 00 00 01 00 00 00 00 00 00 00 00 00

C2 43 D5 E2 00

00 00 00 01 00 00 00 00 00 00 00 00 00 00 01 02 77

C0 F2 A1 B0 00 00 01 02 77

00 7A 00 00 00 00 01 00 00 00 00 00 00 00 FE E3 DA 74 7F

08 8C 01 0E 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

FB 80 8B 24 75 47 DB +

.....\_CHE

CKPOINTABLE\_OBJE

CT\_GRAPH.... CáJ

μ.(\5GnÆR...v/.A

TTRIBUTES/VARIAB

LE\_VALUE.....Ç

.....d(CáJμ.5R÷

.....TCF.

.....L≥i...w

.....πt△

.....

.....

.....W/Ci\$uG

이동

현재 주소

00000123

메모

끝 주소

00000206

이동

검색

검색 대상

데이터 형식

 8비트 정수 16비트 정수 24비트 정수 32비트 정수 64비트 정수 16비트 부동 소수점 32비트 부동 소수점 64비트 부동 소수점 LEB128 16진수 값 텍스트

텍스트 인코딩

 모든 인코딩 역슬래시 변환 대/소문자 구분 (빠르게)

대/소문자 구분

 바이트 순서 리틀 엔디안 빅 엔디안 모두 찾기 찾아 바꾸기

검색 형태

다음 찾기

Output:

2~752020  
745873449

데이터 검사 (빅 엔디안)



새 파일



파일 열기



다른 이름으로 저장



실행 취소



다시 실행



도구



번역



설정



도움말

PATREON

PayPal

## 파일 정보

## -제목 없음- ×

## variables.index ×

파일 이름

variables.index

000000000

파일 크기

207바이트

00000016

## 데이터 검사 (리틀 엔디안)

00000048

## 형식

부호없는  
(Unsigned) (+)부호있는 (Signed)  
(±)

00000064

8비트 정수

194

-62

00000080

16비트 정수

17346

17346

00000112

24비트 정수

13976514

-2800702

00000096

32비트 정수

3805627330

-489339966

00000144

64비트 정수 (+)

3805627330

00000160

64비트 정수 (±)

3805627330

00000176

16비트 부동 소수점

3.878906

00000192

32비트 부동 소수점

-1.9670195e+21

00000000

64비트 부동 소수점

1.880229724627549e-314

00000000

LEB128 (+)

8642

00000000

LEB128 (±)

-7742

00000000

MS-DOS 날짜, 시간

2093-06-21 08:30:04 Local

00000000

OLE 2.0 날짜, 시간

1899-12-30 00:00:00.000 UTC

00000000

UNIX 32비트 날짜, 시간

2090-08-05 14:42:10 UTC

00000000

Macintosh HFS 날짜,  
시간

2024-08-04 23:42:10 Local

00000000

Macintosh HFS+ 날짜,  
시간

2024-08-04 14:42:10 UTC

00000000

이진

● ● ○ ○ ○ ○ ○ ○

00000000

데이터 검사 (빅 엔디안)

+

00	00	06	08	01	1A	02	08	01	00	1C	11	5F	43	48	45
43	4B	50	4F	49	4E	54	41	42	4C	45	5F	4F	42	4A	45
43	54	5F	47	52	41	50	48	08	07	12	00	20	80	A0	D9
E6	1D	28	5C	35	47	FC	92	52	00	1C	1A	76	2F	2E	41
54	54	52	49	42	55	54	45	53	2F	56	41	52	49	41	42
4C	45	5F	56	41	4C	55	45	08	01	12	0B	12	05	08	80
DA	C4	09	12	02	08	64	28	80	A0	D9	E6	1D	35	52	F6
0E	11	00	00	00	00	01	00	00	00	C2	43	D5	E2	00	00

.....	.....	_CHE
CKPOINTABLE_OBJE	.....	
CT_GRAPH....	.....	ÇáJ
µ.(\5GnÆR...v/.A	.....	
TTRIBUTES/VARIAB	.....	
LE_VALUE.....	.....	Ç
.....d(ÇáJµ.5R÷	.....	TCF.
.....L≥i  ...w	.....	
.....z.....πt△	.....	
.....í.....	.....	
.....	.....	
.....W/Çi\$uG	.....	

이동

현재 주소 00000123

끝 주소 00000206

이동

## 검색

검색 대상

데이터 형식

- 8비트 정수
- 16비트 정수
- 24비트 정수
- 32비트 정수
- 64비트 정수
- 16비트 부동 소수점
- 32비트 부동 소수점
- 64비트 부동 소수점
- LEB128
- 16진수 값
- 텍스트

텍스트 인코딩

모든 인코딩

- 역슬래시 변환
- 대/소문자 구분 (빠르게)
- 리틀 엔디안
- 빅 엔디안
- 모두 찾기
- 찾아 바꾸기

대/소문자 구분

바이트 순서

검색 형태

다음 찾기

체크섬 -> 마스킹 해당 부분까지 다시  
체크섬 -> 마스킹 하면 나옵니다.

- Import crc32c
- crc32c.crc32c(open("variables.index","rb").read(123))

```
>>>  
>>>  
>>>  
>>> crc32c.crc32c(open("variables.index_old","rb").read(123))  
745873449  
>>>  
>>>  
>>> █
```



새 파일



파일 열기



다른 이름으로 저장



실행 취소



다시 실행



도구



번역



설정



도움말

PATREON

PayPal

## 파일 정보

## -제목 없음- ×

## variables.index ×

파일 이름

variables.index

000000000

파일 크기

207바이트

00000016

## 데이터 검사 (리틀 엔디안)

00000048

## 형식

부호없는  
(Unsigned) (+)부호있는 (Signed)  
(±)

00000080

8비트 정수

194

-62

00000096

16비트 정수

17346

17346

00000112

24비트 정수

13976514

-2800702

0E 11 00 00 00 00 01 00 00 00 00 C2 43 D5 E2 00

32비트 정수

3805627330

-489339966

00000144

64비트 정수 (+)

3805627330

00000160

64비트 정수 (±)

3805627330

00000176

16비트 부동 소수점

3.878906

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

32비트 부동 소수점

-1.9670195e+21

C0 F2 A1 B0 00 01 02 77

64비트 부동 소수점

1.880229724627549e-314

00 7A 00 00 00 00 01 00 00 00 00 FE E3 DA 74 7F

LEB128 (+)

8642

08 8C 01 0E 00 00 00 00 00 00 00 00 00 00 00 00

LEB128 (±)

-7742

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

MS-DOS 날짜, 시간

2093-06-21 08:30:04 Local

FB 80 8B 24 75 47 DB +

OLE 2.0 날짜, 시간

1899-12-30 00:00:00.000 UTC

W/Çü\$uG

UNIX 32비트 날짜, 시간

2090-08-05 14:42:10 UTC

Macintosh HFS 날짜,  
시간

2024-08-04 23:42:10 Local

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Macintosh HFS+ 날짜,  
시간

2024-08-04 14:42:10 UTC

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

이진

()

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

데이터 검사 (빅 엔디안)

+

## 이동

현재 주소

00000123

메모

끝 주소

00000206

이동

## 검색

검색 대상

8비트 정수

16비트 정수

24비트 정수

32비트 정수

64비트 정수

16비트 부동 소수점

32비트 부동 소수점

64비트 부동 소수점

LEB128

16진수 값

텍스트

모든 인코딩

역슬래시 변환

대/소문자 구분 (빠르게)

바이트 순서

리틀 엔디안

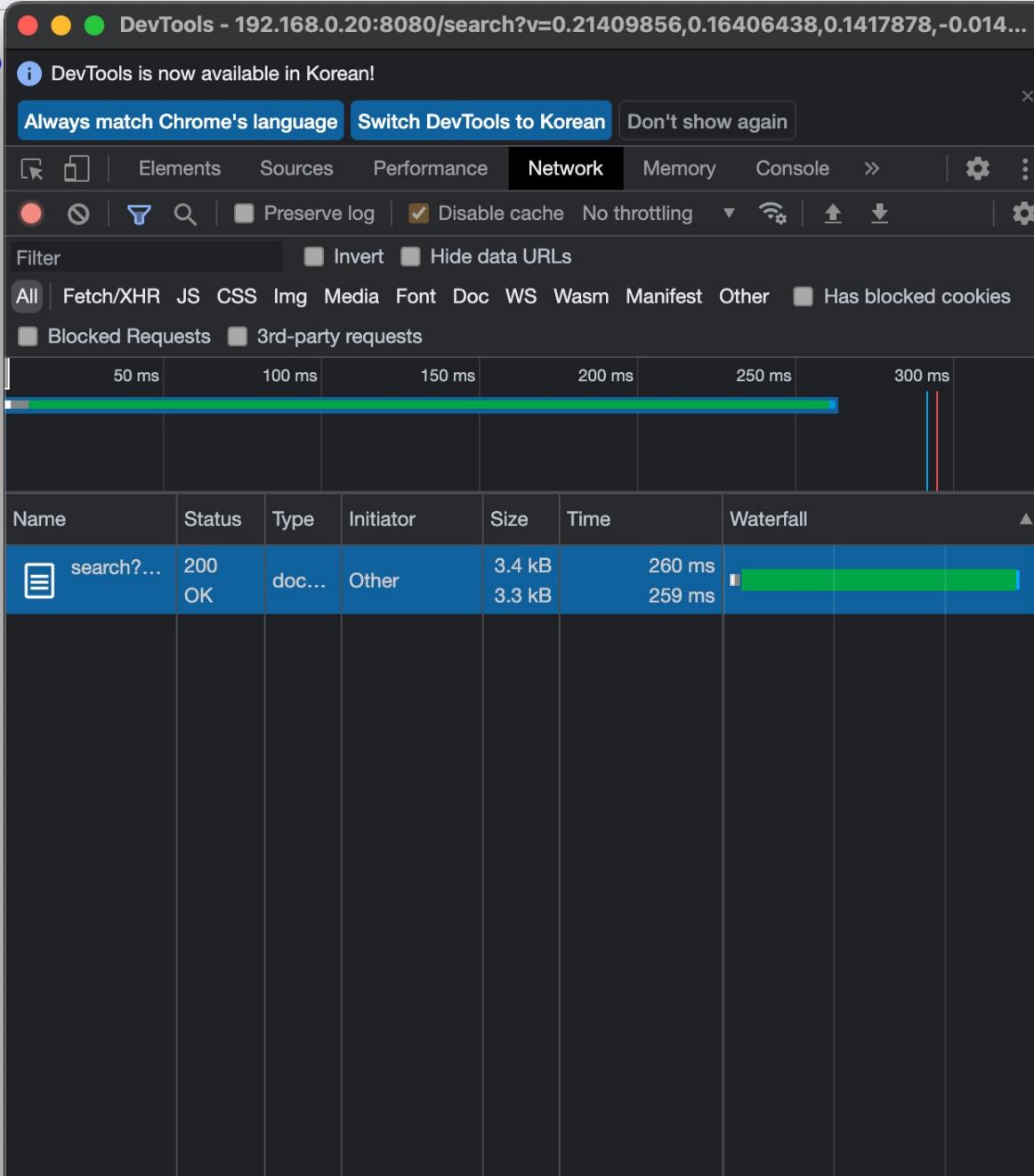
빅 엔디안

모두 찾기

찾아 바꾸기

다음 찾기

```
{  
    candidate: 20000000  
    + input: [ ... ],  
    - vector: [  
        - [  
            0.2292539,  
            8485267  
        ],  
        - [  
            0.22172275,  
            10990239  
        ],  
        - [  
            0.22162317,  
            9830895  
        ],  
        - [  
            0.2163642,  
            2037958  
        ],  
        - [  
            0.21576841,  
            565431  
        ],  
        - [  
            0.21571639,  
            13044944  
        ],  
        - [  
            0.21198863,  
            4705227  
        ],  
        - [  
            0.2092832,  
            6484349  
        ],  
        - [  
            0.20698527,  
            16940611  
        ],  
        - [  
            0.20638,  
            16940611  
        ]  
    ]  
}
```



# 결과

- feature extraction 그래프와 stored embeddings 을 결합 하였습니다.
- 기존에는 동일한 계산그래프를 사용할 때 placeholder를 사용하며 뮤시적 메모리 복사를 통한 오버헤드 존재 -> 해결
- 기존 CPU 대비 25배 속도 증가
- 기존 GPU 대비 4배 증가
- 1억 수준의 임베딩 유사도 계산에 대해서 1초 미만 결과 내는게 가능해짐
- 자바나 스칼라 코틀린 코드에서 numpy를 쓰지 않아도 됨

# 참고 자료

- <https://github.com/tensorflow/tensorflow/issues/51870#>
- <https://stackoverflow.com/questions/51244489/valueerror-cannot-create-a-tensor-proto-whose-content-is-larger-than-2gb>
- <https://stackoverflow.com/questions/35394103/initializing-tensorflow-variable-with-an-array-larger-than-2gb>
- <http://ostack.cn/?qa=604286/>