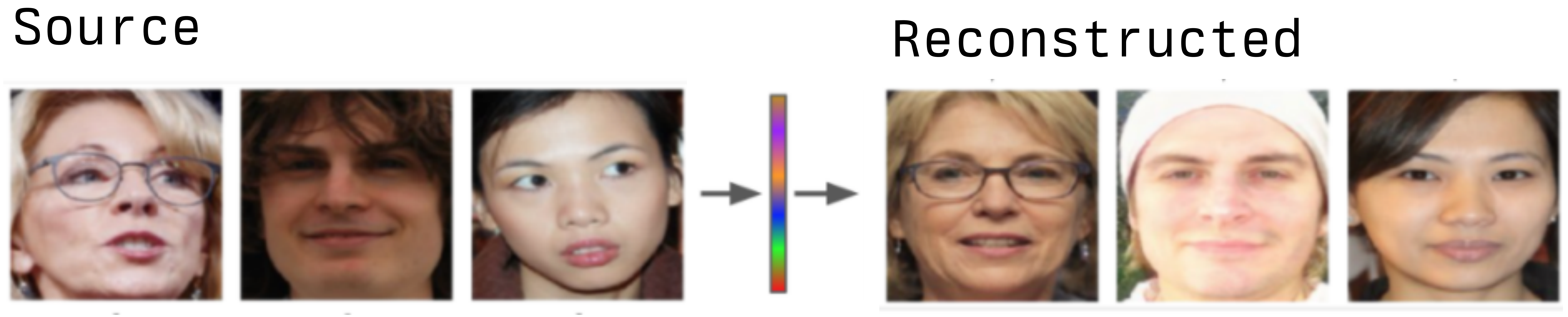
Salty Embeddings

Privacy Preserving RAG on MongoDB Atlas

Vitaly Kleban, San-Francisco, 2024
averagejoe.ai

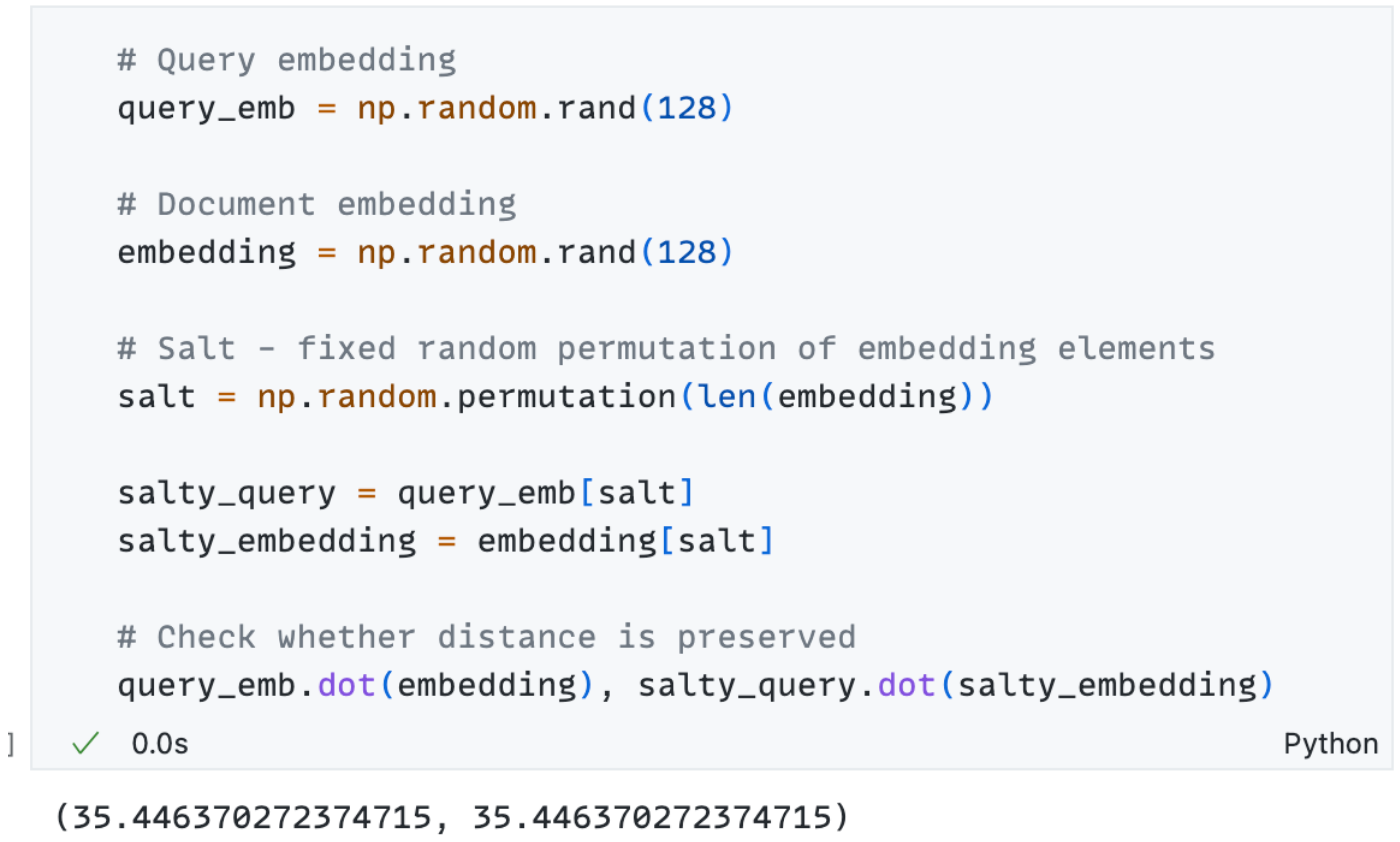
$$\texttt{sha512(password)} \longrightarrow \texttt{sha512(}\overbrace{\boxed{\texttt{password + }\textbf{salt}}}^{\text{salt prevents brute-force attacks}}\texttt{)}$$

$$\texttt{embeddings} \longrightarrow \texttt{???}$$



client side encryption prevents data leakage
queryable encryption makes search efficient

$$\{data,\ embedding\} \longrightarrow \{\boxed{encrypt(data)},\ \boxed{embedding}\}$$



```
# Query embedding
query_emb = np.random.rand(128)

# Document embedding
embedding = np.random.rand(128)

# Salt - fixed random permutation of embedding elements
salt = np.random.permutation(len(embedding))

salty_query = query_emb[salt]
salty_embedding = embedding[salt]

# Check whether distance is preserved
query_emb.dot(embedding), salty_query.dot(salty_embedding)
```

```
(35.446370272374715, 35.446370272374715)
```

“Standard” encryption destroys distance between vectors and prevents vector search

Our options:
homomorphic encryption,
secure multiparty computation,
locality-sensitive hashing,
functional encryption,
random projection, permutation

Client side encryption 🔒

+

Queryable encryption 🔍

+

Salty embeddings 🧂

=

Privacy Preserving RAG 🛡️