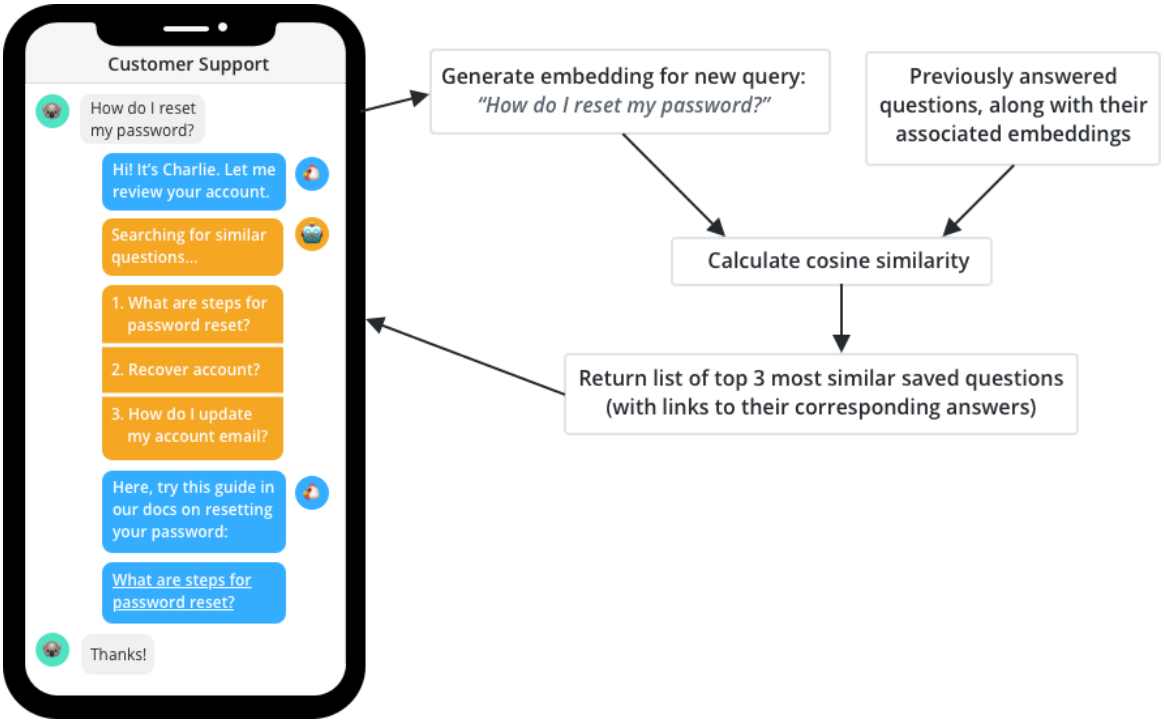
17 JANUARY 2019 / COMMUNITY

# Using NLP to Automate Customer Support, Part Two

Let's build a natural language processing (NLP) model that can help out your customer support agents by suggesting previously-asked, similar questions.

In our last post, we learned how to use one-hot encodings and word embeddings for various NLP tasks. We also demonstrated how approaches such as `word2vec` use *context* to train models that create embeddings with rich semantic meaning.

That *rich semantic meaning* is exactly what we're after with our customer support automation project. In this post, we'll build an NLP model that can help us respond to the nearly infinite ways customers might be asking us the same or similar questions. Here's a quick recap of our customer support NLP model in action:




Customer support suggestions, powered by universal sentence embeddings

To do this, we're going to learn about **universal sentence embeddings**, and then build an NLP model that uses universal sentence embeddings to group customer support questions by their semantic similarity.

# Sentence embeddings: the next frontier

Our customers are (typically) using complete sentences in their

support questions. We'd like to be able to handle the full semantic meaning of these customer support requests, including every little nuanced word choice that they make.

**Universal Sentence Embeddings (USE)** promise to address all of the issues of word embeddings by capturing the semantic meaning of a sentence and identifying order. They're considered "universal" because they promise to encode general properties of sentences due to being trained on very large datasets. In theory, they can then be applied to any downstream NLP task without requiring domain specific knowledge.

With USEs, all we need to do is pass a sentence into the model and we'll be able to get back an embedding representing the entire sentence. That sounds easy enough, right?

## The complexities of sentences

As you might expect, it turns out that sentences are much more complex than words. Some estimates state that there are up to 500,000 words in the English language. And that's not even including emojis 🤔 . Assuming that we can combine these words in any order for sentences up to 100 words in length means – well, it means that there's a whole lot of potential sentences out there in the world!

Now, many of these potential sentences are gibberish – but this itself raises another issue. Namely, what's a valid sentence? Noam Chomsky famously used the sentence *"colorless green ideas sleep furiously"* as an example of a grammatically correct but semantically nonsense sentence. We don't need to get into the heated debate of what is or is not a valid sentence, but let's all agree – it's a complicated mess.

As we might expect, it's going to be more difficult to capture all the context we need for sentences in a fixed dimensional embedding. We're going to need two things:

1. Many more data
2. Many more dimensions to represent this extra context

Luckily, we can let the folks over at Google worry about these data requirements and use their pre-trained model for sentence embedding.

## Pre-trained model

We'll be using their pre-trained USE model which produces sentence embeddings of 512 dimensions. While it's indeed larger than word embeddings (typically 100-300 dimensions), it's not quite as large as we feared.

In this model, the size of the embeddings for the USE was chosen so that the model could be as generic as possible. The data it was trained upon was rather general - including supervised and unsupervised training data such as Wikipedia, news data, and question-response type dialogues.

If you want a deeper insight on the training and the underlying neural network architecture, take a look at their paper.

# Generating our first sentence embedding

Let's generate an embedding to represent a test sentence. You can follow along with this example by clicking this button to open JupyterLab workspace on FloydHub. Then just run the `GenerateEmbedding` notebook.

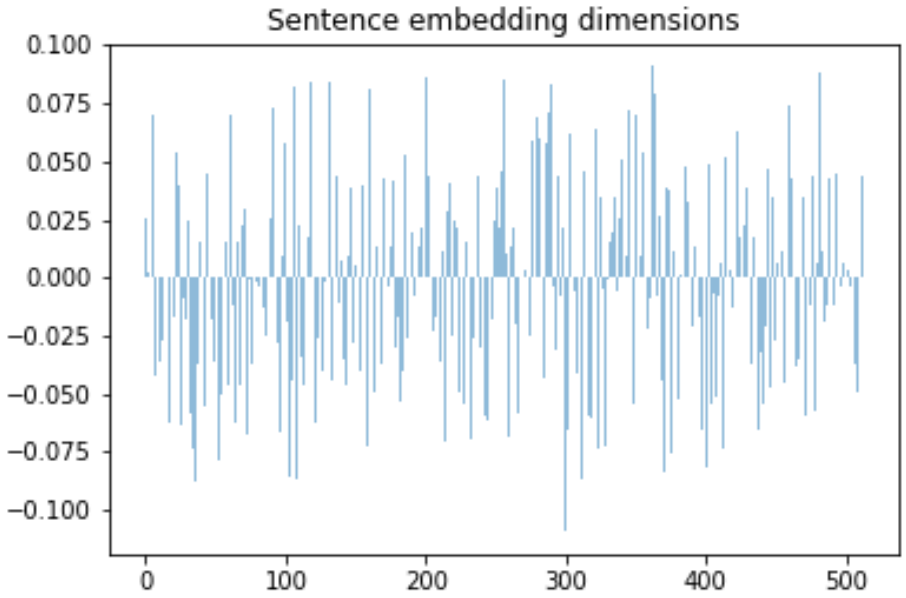
Run on FloydHub

To start with, let's generate an embedding for a sentence such as `how can I reset my password.`

```
# Import libraries
import tensorflow as tf
import tensorflow_hub as hub
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt; plt.rcdefaults()

# Download the USE module
module_url = "https://tfhub.dev/google/universal-sentence-encoder-
embed = hub.Module(module_url)

# Generate the embedding and print out some descriptive data
def generate_embedding(messages):
    with tf.Session() as session:
      session.run([tf.global_variables_initializer(), tf.tables_in
      message_embeddings = session.run(embed(messages))
      for i, message_embedding in enumerate(np.array(message_embed
        print("Message: {}".format(messages[i]))
        print("Embedding size: {}".format(len(message_embedding)))
        message_embedding_snippet = ", ".join(
            (str(x) for x in message_embedding[:3]))
        print("Embedding: [{}, ...]\n".format(message_embedding_sn
    return(message_embeddings[0])

emb = generate_embedding(["How can I reset my password"])

# visualize it in a barchart to show the range of values in the 51
y_pos = np.arange(len(emb))
plt.bar(y_pos, emb, align='center', alpha=0.5)
```

```
plt.title('Sentence embedding dimensions')
plt.show()
```

This outputs the rather unwieldy 512 array that looks something like this:




This shows the range and distribution of values for our sentence embedding

Looking at the output alone, it's rather difficult to evaluate the embedding. But, as we saw with word embeddings in our previous post, let's try to look at the dimensions of these embeddings to see if we can detect any patterns.

For example, let's start with our above sentence `How can I reset my password` . We have an embedding for that already. Next, let's make some manual changes to this original sentence so that we can create a list of sentences with a clear range of similarity.

> What if we change one word slightly, or change the ordering or the form of the question? Do we notice certain dimensions changing or does it appear random? Is is like word embeddings and we can see some pattern that seems to correspond to minor changes compared with larger changes?

The table below shows the sentences we will look at together:

| NO | DIFFERENCE | SENTENCE |
|---|---|---|
| 1 | Original sentence | How can I reset my password |
| 2 | Plural word | How can I reset my password**s** |
| 3 | Remove word | ~~**How**~~ can I reset my password |
| 4 | Change order | **reset can password How I my** |
| 5 | Similar | How can I **change** my password |
| 6 | Different sentence | **What is the capital of Ireland** |

To compare the embeddings, we will generate two embeddings. One embedding will represent the original sentence "*How can I reset my password*" and the other will be one of the six altered sentences in

the above table. We want to see what impact the changes to the sentence have on the embeddings.

## Comparing dimensions

If it is a small change, then we would expect the embeddings to change slightly compared to a larger change. By generating two embeddings we can look at the dimensions and see which ones changed and by how much. We can ignore minor changes and look at the dimensions that changed by 10% or more. And we can also get the average change across all dimensions.

## Comparing similarity

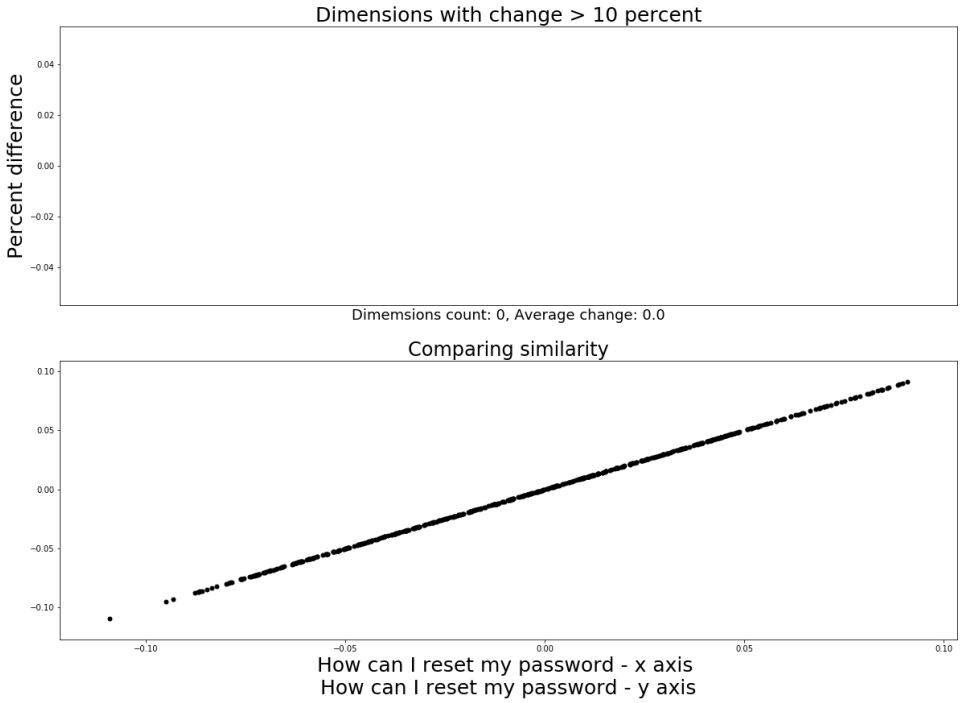
In conjunction with the dimensions, we also want to see how similar the sentence embeddings appear when compared to each other visually. Since we have seen how difficult it is to "eyeball" our embedding, we need to find some other way to visually inspect it.

You might be inclined to just go ahead and start using the embedding due to the difficulty in representing it visually. However, it is well worth spending time on playing around with the output of the model to see if you can infer anything from that. At the very least you will be more comfortable with the output itself when you do use it! This is critical, since we're planning to build a customer-facing chatbot with our sentence embedding model. I'm glad you agree. Let's do this, then!

You can follow along with this example by clicking this button to open JupyterLab workspace on FloydHub. You can replace the test sentences with your own and check out how they compare visually by running the `VisualComparison` notebook.

If we create a scatterplot of any two embeddings, we can see if there is a correlation between the dimensions. For example, if we generate a random distribution, we would expect to see a spherically dispersed arrangement of points. Conversely, if the dimensions correlate completely, then it should be a straight line like we would find in a linear regression example.

To show this, let's compare two identical sentence embeddings.



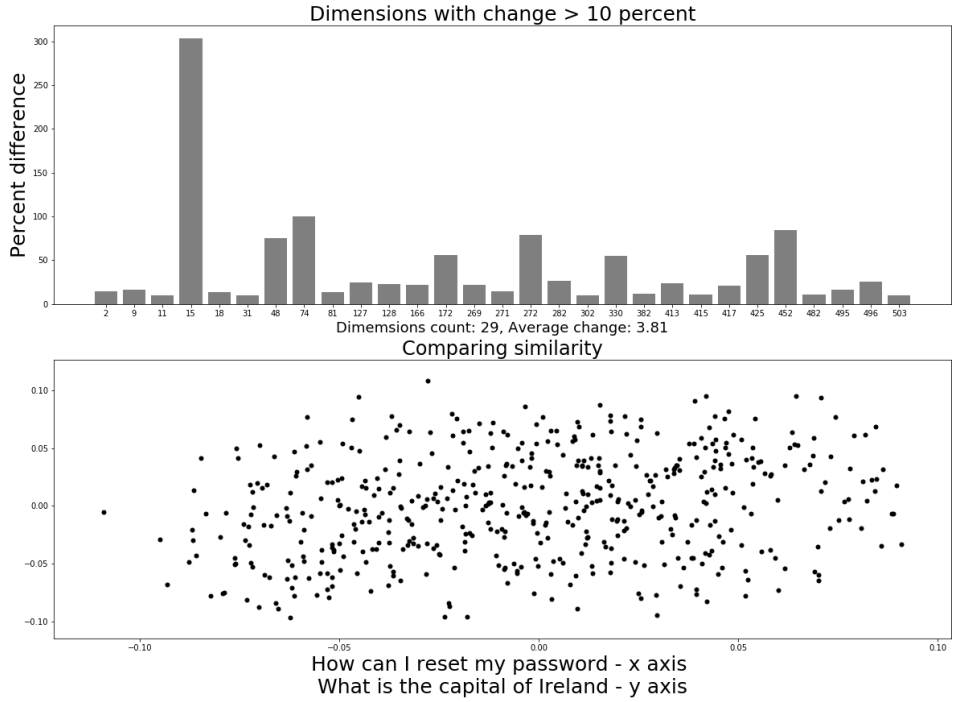

These plots show that the two embeddings which were generated are identical(which makes sense as the two sentences used are the same)

In the above diagram we are comparing the same sentence, "*How can I reset my password*", but we are generating two embeddings for it. Think of this as a way to make sure that the embeddings are

consistent. If there was a difference here then we would be slightly worried about the model.

As you can see the dimensions are the same, there is no difference between the dimensions from the embeddings. And in the similarity scatterplot we get an almost perfect straight line. Again, this is just created by taking each dimensions and mapping it against the corresponding dimensions.

As a comparison, let's now look at two completely different sentences and see how they compare to make sure the way we are comparing them visually makes some sense.




In contrast to the previous diagram these plots show how two embeddings, generated from very different sentences, compare visually.

We are now comparing the sentences "*How can I reset my*

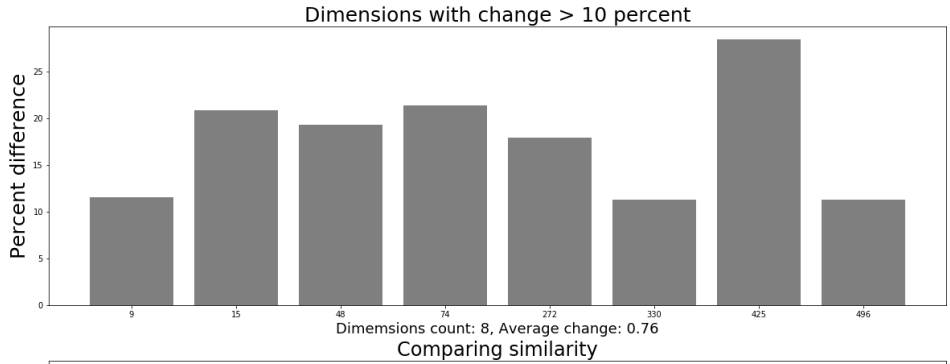
*password*" with "*What is the capital of Ireland*". We can clearly see the difference with our diagrams. The average difference between dimensions is 3.81. We can see a number of dimensions changed by over 10% which makes sense as they are difference sentences.

What is interesting in the dimension bar chart is that one dimensions, number `15`, changed by much more than the others.

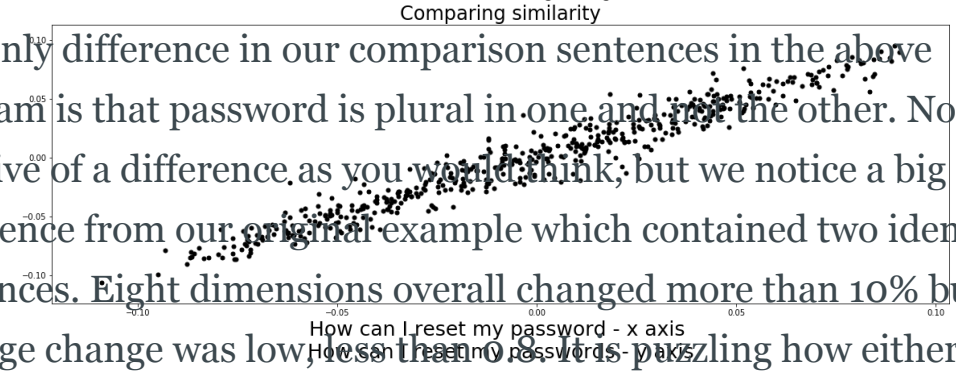
> Does this represent a dimension which grades overall similarity? Or measures word comparison? Or does it identify sentence categories?

We can theorize what it means, but it clearly seems to be a stronger signal indicator that any of the other dimensions. The scatterplot also indicates a random spherical dispersion which seems correct as the sentences are completely unrelated. This seems to verify that there is some merit to our chosen methods to represent these sentences.

Now let's look at some sentences which should be pretty similar from a semantic perspective:



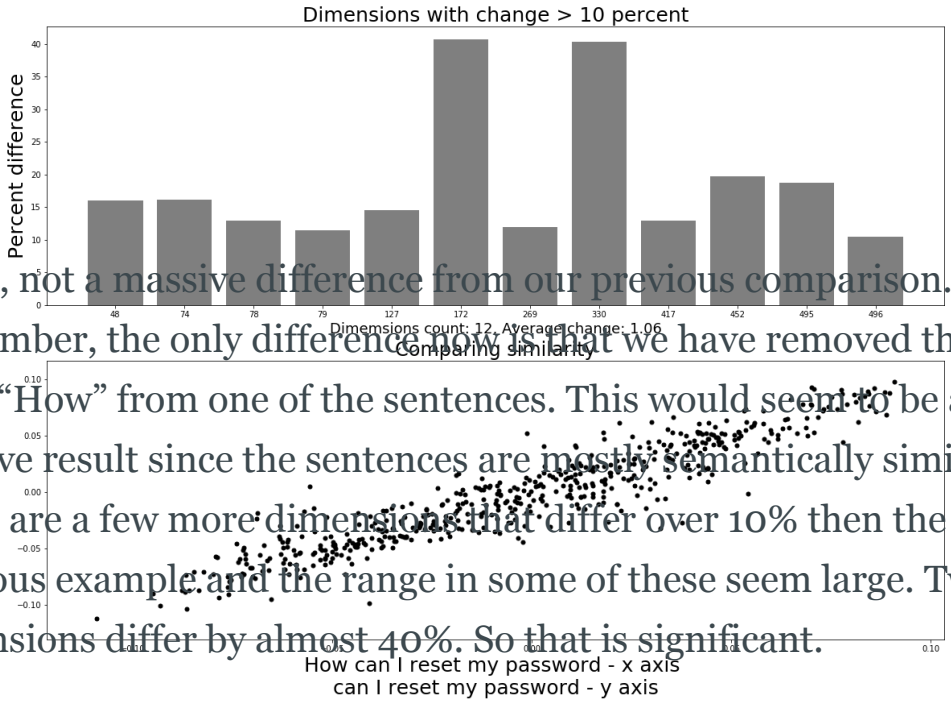

Dimemsions count: 8, Average change: 0.76




The only difference in our comparison sentences in the above diagram is that password is plural in one and not the other. Not as massive of a difference as you would think, but we notice a big difference from our original example which contained two identical sentences. Eight dimensions overall changed more than 10% but the average change was low, less than 0.8. It is puzzling how either dimensions could represent one plural word change. Our scatterplot shows a strong correlation and we can clearly see that the sentence is similar from this plot.

We only made a minor change to one word yet we see that it had an impact and differs from significantly on a number of dimensions from our earlier identical example.

Let's next leave out a word and see how they compare.




Again, not a massive difference from our previous comparison. Remember, the only difference now is that we have removed the word "How" from one of the sentences. This would seem to be a positive result since the sentences are mostly semantically similar. There are a few more dimensions that differ over 10% then the previous example and the range in some of these seem large. Two dimensions differ by almost 40%. So that is significant.

How does removing a word impact the similarity? The semantic meaning is the same and yet we see a scatter plot that still shows a relatively well aligned pattern

Again, it is difficult to know how a small change is represented by the subsequent change in dimensions.

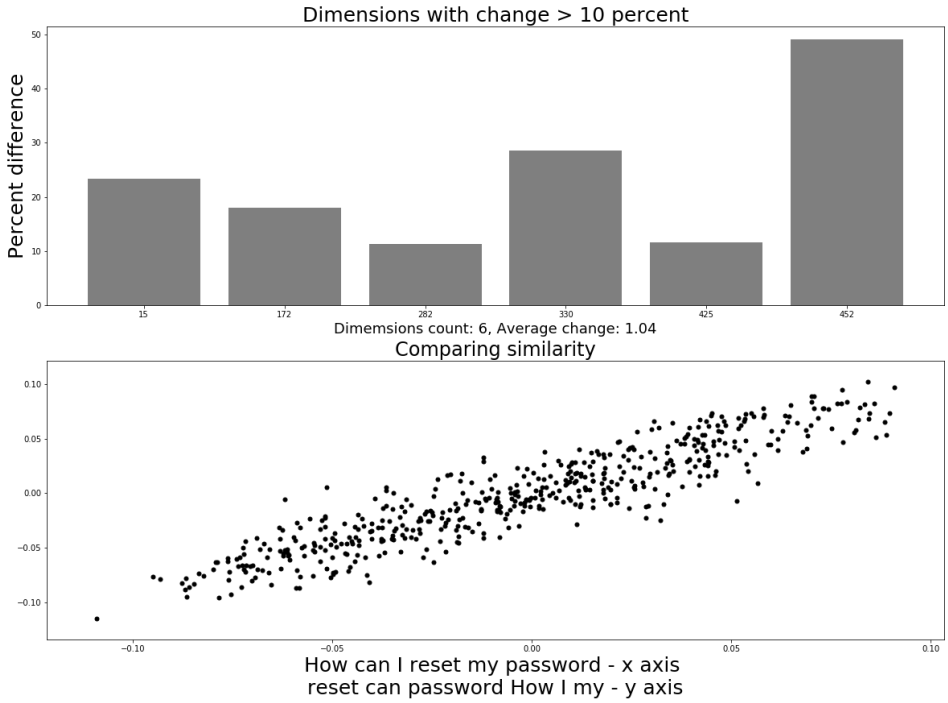
> Are the two dimensions that changed the most identifying sentence length or word count for example?

Both of these dimensions have changed by more than 10% in our previous examples, but not always together.

So it appears difficult to understand these changes fully. It could be that we simply cannot understand the complex relationship and features the deep learning network learned for these embeddings.

> Maybe the model represents sentences in an entirely different way to how humans understand sentences?

For our penultimate example, let's look at whether a change in ordering can be identified from the embeddings:




These plots show how the sentence embeddings capture ordering information. The sentences are identical but our of order and we see a clear difference between the earlier identical plot.

Firstly, it is clear the change in ordering is identified. If it were not we would see graphs similar to our first (identical) example since all the words are identical, just out of order. Only six dimensions changed by over 10% so that is much less than any of the other examples.

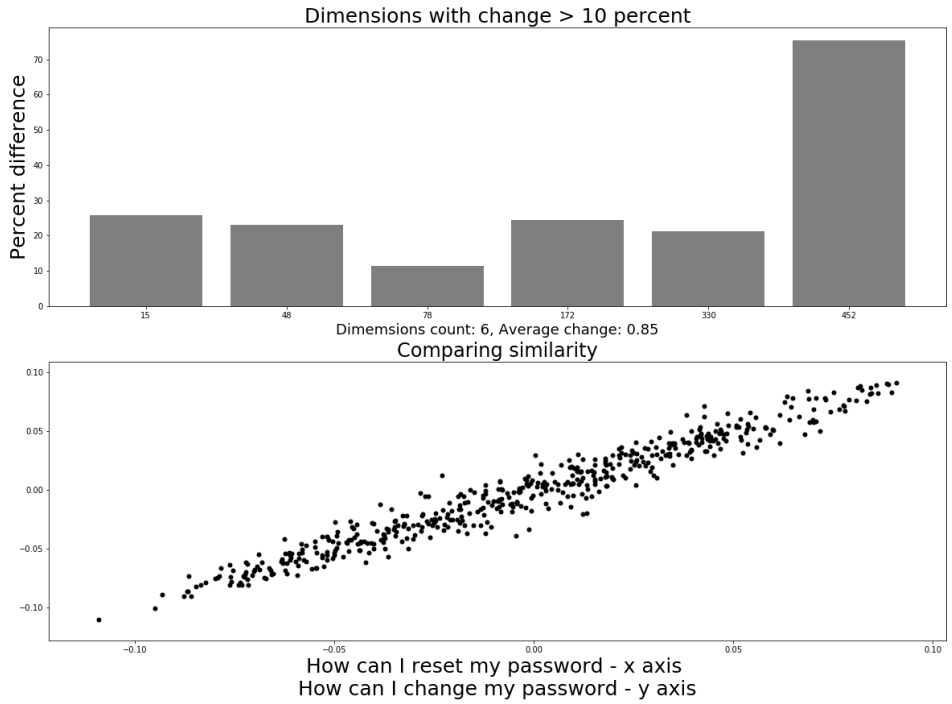
> Do these dimensions represent order?

It seems unlikely since none of them are unique, they have changed in the other examples also, but not by as much. Dimension 452, for example, changed in the previous example by nearly 30% but here it

changed by almost 50%. If one dimension changed by 70% here and no other dimension changed then we could see that as the "order" dimension but it does not seem that simple.

As we noted earlier, being able to identify ordering is an important feature since this is something that is very difficult to capture using word embeddings. When we jumble up words they appear identical using these models. But we can see that sentence embeddings capture this ordering feature.

Finally, let's look at what happens when we change a word but maintain a similar semantic context.




What happens when we use different words that mean the same thing? Reset and change are different words but can be used in a similar way. The plots seem to indicate that this similarity is captured in the embeddings.

We have replaced “reset” with “change” so the sentence is quite similar. We would characterize them as being almost identical and in nearly every support case would handle them as if they were the same query. And this does appear to be the case here. Only six dimensions changed and the average change was 0.85. This is slightly more that the change for the plural difference and slightly less than our ordering and word removal examples.

## So what have we learned from visually inspecting our model output?

Sentence embedding definitely warrants more investigation.

They capture semantic similarity and also ordering. We can see clear differences between identical sentences and completely unrelated sentences. We cannot identify a clear pattern of change in the dimensions that differ between examples. There seem to be a small number of dimensions that change in each example. This would seem to indicate that most of the information is being carried by only a few dimensions.

But we need to be careful not to draw too many conclusions from our brief investigation. The example sentences were carefully selected and do not really test how well sentence embeddings would perform in a real scenario.

Next, we need to manually curate a list of questions and measure their similarity **using something other than visual inspection**.

# Create a sentence embedding baseline

We now want to understand the range and accuracy of our sentence embeddings.

To do this, we need to create a baseline set of sentences pairs. The baseline will contain sentence pairs that have been altered to range from **mostly similar** to **almost completely unrelated sentences**. We can use this baseline to see if the measuring similarity between our embeddings correctly reflect the semantic disparity between out sentences.

For example, the table below shows an example sentence pair with related altered sentences and what we would expect in terms of their similarity.

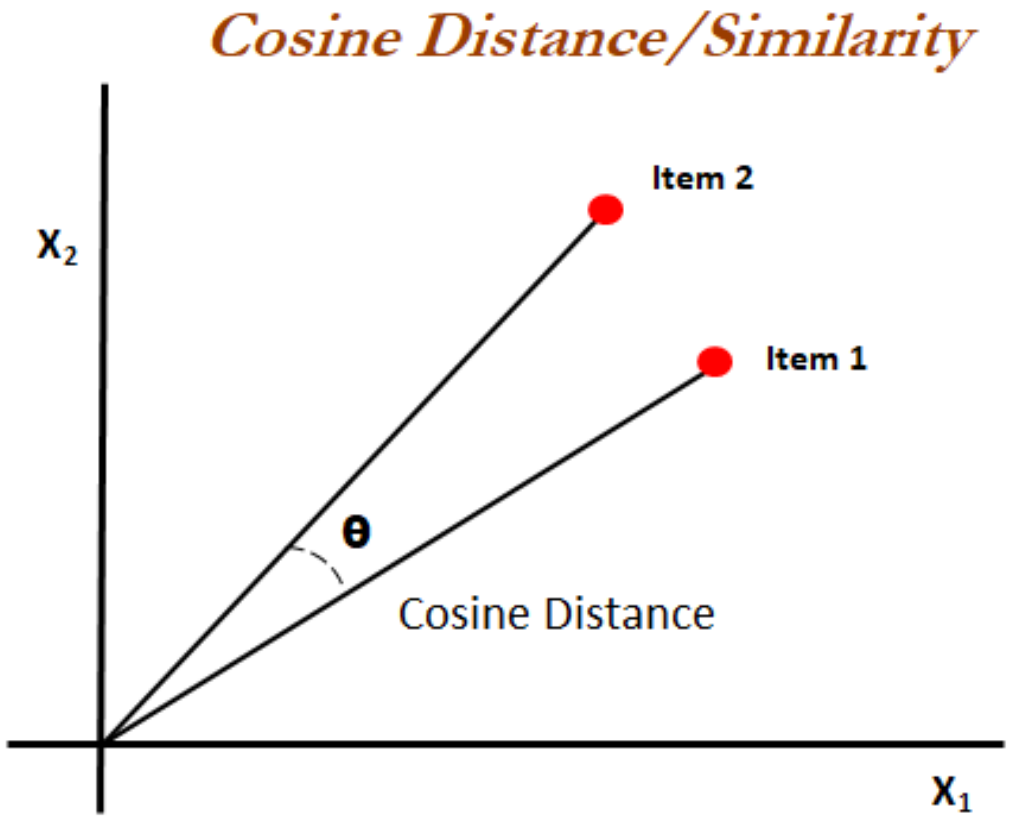
| NO. | COMPARISON SENTENCE | ORIGINAL SENTENCE | EXPECT |
|---|---|---|---|
| 1 | How can I rename a file? | How can I rename a file? | A perfe |
| 2 | How can I change a file? | How can I rename a file? | A close |
| 3 | Can I change the name of a file ? | How can I rename a file? | A some |
| 4 | Can rename How file I a? | How can I rename a file? | Less of |
| 5 | How can I rename a city? | How can I rename a file? | Should |
| 6 | My cats name is Mr Jingles | How can I rename a file? | No simi |

Ideally we would like a clear separation between these sentences as the become progressively semantically different. If our scoring system is unable to differentiate between the extremes of these examples (i.e. 1 and 6) then it will be of little use.

There is some ambiguity between sentences 2-4. It is difficult to know how similar scores should be between these sentences. We

know they are semantically similar but how much of that semantic similarity can be captured in our embeddings is difficult to know. Sentence 5 is an interesting alteration where we only changed one word but this has a significant impact on the overall semantic meaning of the sentence. You may think that this is an unfair comparison as it is a somewhat contrived example. However, it is a good example of where we may get incorrect results when looking for similarity with embeddings and it also show the extra level of complexity when compared with word embeddings.

## Cosine similarity




Since our embeddings are vectors we can measure similarity using their magnitude and direction

Remember that our embeddings are vectors, so we can use this fact to evaluate how similar they are to each other. Cosine similarity is a common way to measure similarity by getting the cosine of the angle between our two vectors, which in this case represent our sentences. Cosine similarity is the method used to measure similarity in the original USE paper and also the method used in the code examples provided with the TensorFlow USE module. We can use this method to measure the similarity of our sentences.

## Dataset

We noted earlier that we will manually curate the sentences we use in our baseline. We still, however, need a dataset of sentences that we use as the original sentences. We want to ensure there are a variety of base sentences to start with and that the overall number of sentences is large. If we have a small number of sentences it would be easier to "game" our test since there would not be much choice. Ensuring a large number of potential matches is a better simulation of a real life situation.

One dataset that seems perfect for our use case is the Quora duplicate sentences dataset available on Kaggle. This is a set of

sentence pairs and a corresponding label to identify if these sentences are indeed duplicates.

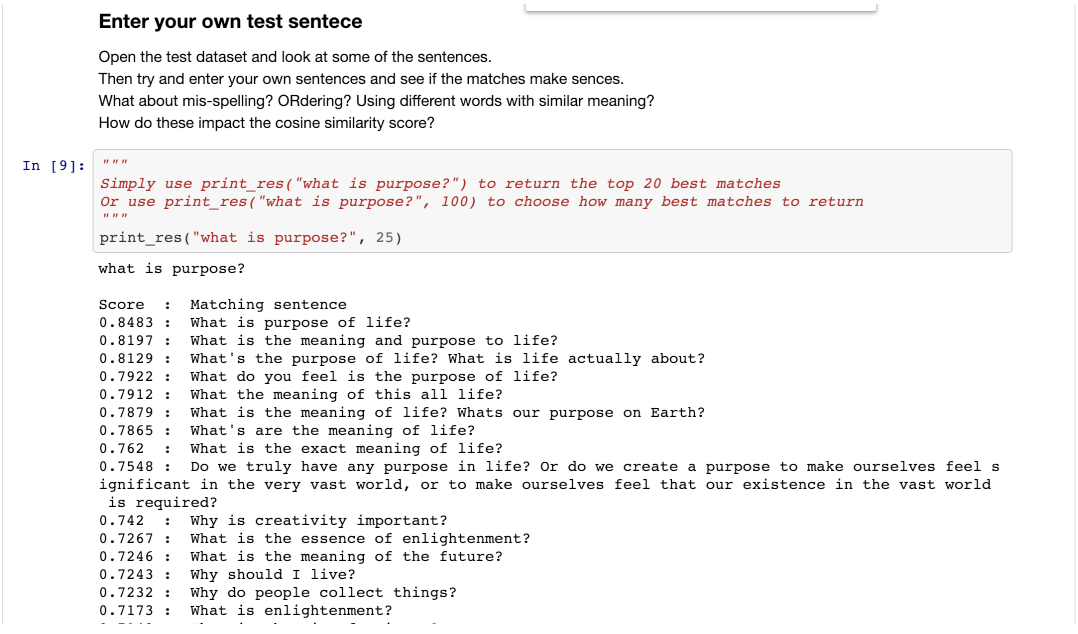
## Test the dataset yourself

So far we have generated some embeddings and visually analyzed a series of sentence pairs to get a feel for how the USE works. However, the best way to do this is to take it for a test drive yourself.

To do this, we've already generated the embeddings for the Quora dataset sentences. These have been saved to the GitHub repo using pickle. This way you only need to generate the embeddings for you test sentence and then you can look at the best matches returned via cosine similarity with the Quora questions.

To do this, simply run the `TestSentences` notebook and follow the instructions to add your own sentence.





**Enter your own test sentece**

Open the test dataset and look at some of the sentences.
Then try and enter your own sentences and see if the matches make sences.
What about mis-spelling? ORdering? Using different words with similar meaning?
How do these impact the cosine similarity score?

In [9]:
```
"""
Simply use print_res("what is purpose?") to return the top 20 best matches
Or use print_res("what is purpose?", 100) to choose how many best matches to return
"""
print_res("what is purpose?", 25)
```

```
what is purpose?

Score  :  Matching sentence
0.8483 :  What is purpose of life?
0.8197 :  What is the meaning and purpose to life?
0.8129 :  What's the purpose of life? What is life actually about?
0.7922 :  What do you feel is the purpose of life?
0.7912 :  What the meaning of this all life?
0.7879 :  What is the meaning of life? Whats our purpose on Earth?
0.7865 :  What's are the meaning of life?
0.762  :  What is the exact meaning of life?
0.7548 :  Do we truly have any purpose in life? Or do we create a purpose to make ourselves feel s
ignificant in the very vast world, or to make ourselves feel that our existence in the vast world
 is required?
0.742  :  Why is creativity important?
0.7267 :  What is the essence of enlightenment?
0.7246 :  What is the meaning of the future?
0.7243 :  Why should I live?
0.7232 :  Why do people collect things?
0.7173 :  What is enlightenment?
```

Open the notebook and add your own sentence and run this cell to see what matches

## Back to our baseline

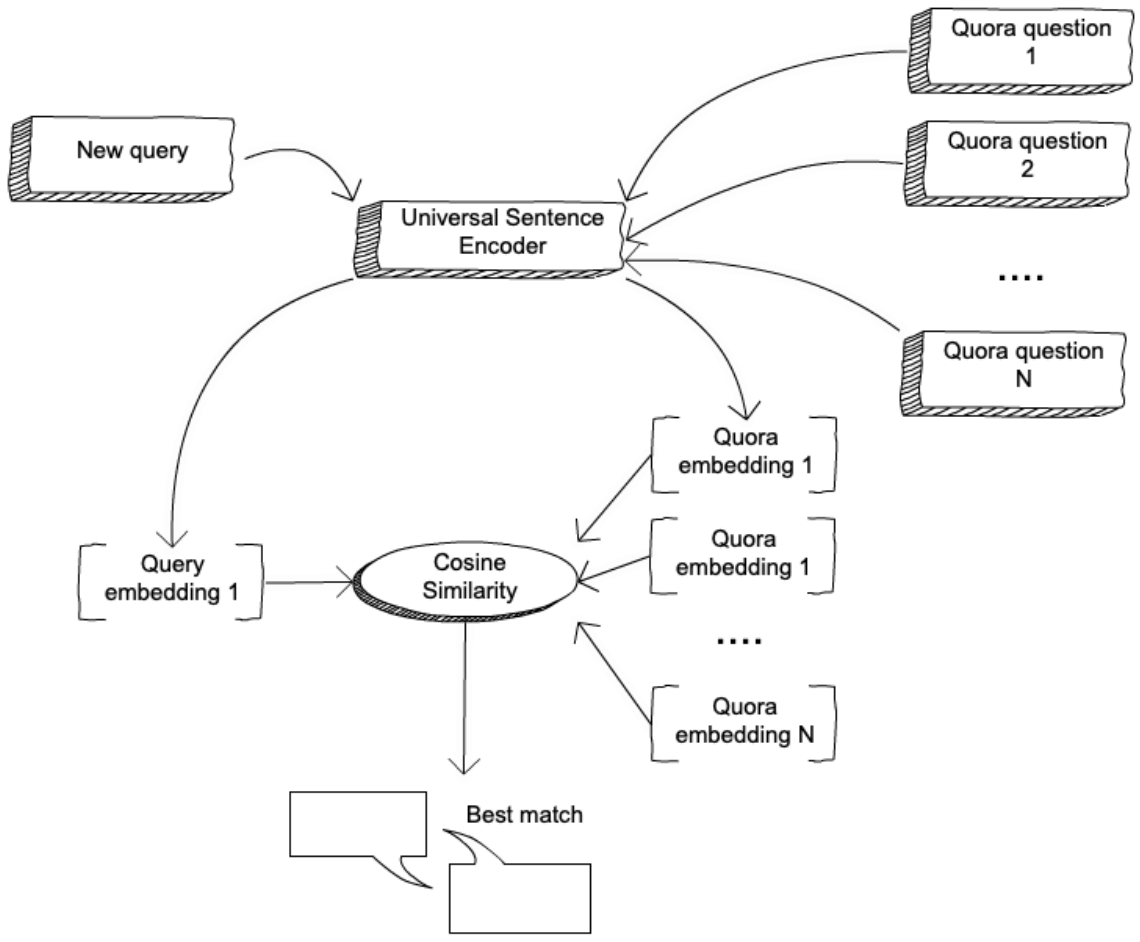
For ease of use, let's just select the first 1,000 sentence pairs and create a baseline of 35 sentences from 6 original sentences. You can see the list of sentences in our baseline below, the original sentences are in bold and the other sentences are the manually altered versions of the originals.

| QUESTION |
| --- |
| **What is purpose of life?** |
| What is meaning of life? |
| What is purpose of technology? |
| What is porpose of life? |
| What is colour of life? |
| is What purpose of life? |
| **Can I recover my email if I lost the password?** |
| Can I get my email if I misplaced the password? |

| QUESTION |
| --- |
| is it possible to recover my email if I forgot the password? |
| email my password? Can if the I recover forgot I |
| **How do I use Twitter as a business source?** |
| How do I use my cat as a business tool? |
| How do I use Twitter as a business tool? |
| can I use Twitter as a business source? |
| How do I use Twitter as a busines source? |
| Twitter do business I source? How a as use |
| **How do you make a screenshot on a Mac laptop?** |
| How do you make a screenshot on a Windows laptop? |
| How do you take a screenshot on a Mac? |
| How do you take a bcreenshot on a Mac laptop? |
| screenshot on a Mac laptop? |
| you take on do Mac screenshot a a laptop? How |
| **How many months does it take to gain knowledge in developing Android a** |
| How many months does it take to develop Android apps from scratch? |
| How much time does it take to create Android apps from scratch? |
| How many months does it take to gain knowledge in developing apps from scr |
| How many months does it take to gain knowledge in developing Facebook app |
| **When is it too late to learn the piano?** |
| When is it too late to learn the guitar? |

| QUESTION |
| --- |
| Where can I buy a piano? |
| When is it too late to go to work? |
| When is it too late to learn spanish? |
| Where can I learn to play the piano? |
| When is the best time to learn the piano? |
| When is it too late to learn the moonwalk? |

To test this baseline set of questions we will go through the following steps:



Compare each baseline query with the full list of Quora questions and find the best match

This is not the most efficient way to do this. If this were in production we could generate the embeddings for our list of 1,000 potential matches beforehand and only generate the incoming questions as needed. However, this example also shows the speed of generating embeddings when we can generate 1,000 embeddings in a reasonable time for each of our 35 questions. And this includes calculating the cosine similarity also. This shows that from a performance perspective it is reasonably effective.

## Baseline results

The table below shows some of the results results of the baseline evaluation. The columns are as follows:

1. **Query** - These are our baseline questions. They include the original, which should find a match since there is a corresponding duplicate for each of the original questions we chose.
2. **Best Match** - This is the best match from all of the set of 1000 questions based on the highest cosine similarity score.
3. **Original** - The original unaltered Quora sentence for reference
4. **Similarity -** The cosine similarity score calculated representing the best match from the list of potential matches

| QUERY | BEST MAT |
|---|---|

| QUERY | BEST MAT |
|---|---|
| What is meaning of life? | What's a |
| What is purpose of life? | What's th |
| How do I use Twitter as a business tool? | How can |
| How do I use Twitter as a business source? | How can |
| is What purpose of life? | What's th |
| How do you make a screenshot on a Mac laptop? | How do I |
| How much time does it take to create Android apps from scratch? | How muc |
| How do you take a screenshot on a Mac? | How do I |
| is it possible to recover my email if I forgot the password? | I do not r |
| Can I recover my email if I lost the password? | I do not r |
| Can I get my email if I misplaced the password? | I do not r |
| When is it too late to learn the guitar? | I'm almos |
| Twitter do business I source? How a as use | How can |
| How do I use my cat as a business tool? | How do I |
| What is colour of life? | What's a |
| When is it too late to learn the moonwalk? | I'm almos |
| Where can I buy a piano? | How diffi |

## Some observations

- **Overall the results are impressive**. Remember, this is a generally trained model so it has no domain specific training. So to correctly match the duplicated sentences (i.e. it rarely

identifies a pairing that is not labeled as a duplicate) with an "off the shelf" model is significant. **There are strong matches (> 0.8)** even when we altered the sentences with spelling errors or new words to maintain similar semantic meaning.

- **The problem is not with the higher matches but with the lower matches**. If we look at the lowest matched there is only one entry below 0.7. Sentences such as "*How do I use my cat as a business tool?*" and "*What is colour of life?*" found matches with a score of > 0.7. This means there is very little margin between a relatively meaningless sentence and an almost identical match. We can see there is no "weighting" as such for what we might consider semantically important words such as swapping "cat" for "twitter" and changing the context of the entire sentence. Again, this may be an unfair measure since are examples such as this likely to occur in real life.

Regardless, the lack of separation between the top and bottom of our baseline does indicate that we might not be able to confidently use the embeddings to automatically find the best match.

But this does not mean we cannot use our embeddings – far from it. There is clearly a lot of potential to use these embeddings since they do pick up semantic similarity, they identify ordering, and compensate for spelling errors and other minor alterations.

They seem well suited to an internal recommendation system where a top five list of best matches would be returned to a customer support agent. The agent in this case would act as a filter and be able to quickly identify whether any of the recommendations were useful.

**This changes the cosine score from an absolute measure to a relative measure of success or a ranking system.**

For example, if there was a useful recommendation, then the agent can quickly use the previous response to provide an answer to the customer. In this way, the recommendation system saves the agent the bother of having to manually check for previously similar questions. Customers then get quicker responses and new issues can be identified more quickly.

In the next section, we will build a simple prototype for this kind of system.

# Building a customer support recommender

The goal with the recommender system is to allow the user to review what the model considers to be the most similar questions.

We will continue to use our Quora dataset but we can alter it a bit to better represent common queries within your organization which you would like to more easily identify. What we want to end up with is a series of clusters which represent common customer queries.

Queries such as `How can I change my password`, `how can I setup an account`, or `Where do I setup my email` may represent common issues customers encounter and enquire about. There is no need for someone to come up with a new response here, there is probably documentation and a clear series of steps to take to resolve these issues. Instead of your customer support agent spending time looking for a previous similar query, or creating a new response, you would like them to be able to quickly retrieve the responses that were already provided.

To create our example clusters, we can go through the Quora questions and select a sample of queries which look like typical technical support type issues. We want a number of groups so we can pick some queries and then make some manual changes so that there is a cluster of at least 5 queries. Each cluster of queries will be associated with one saved response. The saved response could be a template or a fully completed example, whichever works for you use case. The key is that there is a label or ID for each query to link it to the relevant group.

For our example here we will just create 10 groups of 5 questions to show how this could be setup to generate recommendations for similar queries using sentence embeddings. You will likely have many more questions but it does not change the basic structure of the recommender. The sentence embeddings are quick to generate once you first run it to download the USE module.

If there is such a large number then you can generate the embeddings for your question clusters offline and simply save. Then you would only need to generate the embedding for the new query in production. For now, we'll just generate all the embeddings as needed to keep it simple. The table below shows the example questions we are using for our recommender:

| QUERY |
| --- |
| I forgot my username and have no access to my recovery phone number. How |
| I forgot the password. What should I do to recover my password? |
| What are the steps for a password reset? |
| How do I recover my password when I'm not receiving any recovery email? |

| QUERY |
| --- |
| How can I reset my password? |
| Is there a way to insert multiple images? |
| How do I add an image? |
| Can I change my profile image? |
| Can I upload a image for users? |
| Whats the max size for a profile image? |
| How can I signup online without a credit card? |
| Can I signup without a credit card? |
| Do I need a credit card to start a trial? |
| Do I need a credit cart to setup an account? |
| Do you have a free version? |
| What are some good photo editing apps? |
| Which is best free video editing software? |
| How can I edits videos? |
| Can I update my videos before uploading |
| Are there online tools I can use to share my video |
| How can I upload Folders which contains daily report excel files daily to my usi |
| Can I upload data with a python script? |
| How do I upload a lot of data with a python script? |
| Are there examples of python scripts to upload data? |
| Can i setup a regular upload with a python scrip? |

| QUERY |
| --- |
| How can I search for pending issues that have not been solved yet? |
| How can I seach for open issues? |
| How can I find open issues? |
| Where can i see a list of open and closed issues? |
| Can I search by newest issues? |
| Is there any way to query the metadata? |
| How can I get the metatdata for a query? |
| Is there metadata associated with a query? |
| Where is the metadata stored? |
| Can I search the metadata for a query? |
| How do I update my account? |
| Can I update my account? |
| Why do I need to update my account? |
| When do I need to update my account? |
| Where can I find out how to update my account? |
| How do you disable an account? |
| How do you delete an account? |
| Can I recover a deleted account? |
| Do I lose all my data when I delete my account? |
| Can I archive my account? |
| How do I use API from your website? |

| QUERY |
| --- |
| Is there an API available? |
| Can I use an API? |
| Where can I find info on your API? |
| How do I setup my account to use an API? |

## How to use this model for your own customer support

Sentence embeddings are interesting since they offer a number of ways in which you could potential use them in your business. They are versatile in how they can be deployed. For example, you could build a neural network to identify question and answer pairing. This is similar to the Amazon example we mentioned earlier where questions and answers are encoded and the network learns, via a supervised approach, to match correct pairings. It can then automatically identify an answer for a new incoming query. This is a fairly complex setup but you could use sentence embeddings to encode the questions and answers which would help reduce much of the complexity.

> To simplify it further we are not encoding the answers at all. We are only focusing on the questions. We are comparing the embeddings from new incoming queries with manually selected queries representing common customer issues.

In this way, you use the embeddings to identify the best match between the new query and your saved queries. The saved queries

are linked to a known answer. **Your recommender would be a question-to-question recommender as opposed to a question-to-answer system.** You are then be able to link from the recommended questions to their corresponding answer. The point is that once you understand how to use sentence embeddings you can find different ways to deploy them within your organization. So, while your own use case may be slightly different, you can follow along here and eventually tweak the below example to suit your own needs.

## Test your own sentence recommendations

To generate a similarity, we need to get the cosine score like we did earlier with the baseline questions. When given a new incoming query we can get the cosine similarity between the new query and each saved query We can use the TensorFlow code example on TF-Hub to feed a list of question pairs and get the embeddings and the similarity:

```
# Use a TF placeholder
sts_input1 = tf.placeholder(tf.string, shape=(None))
sts_input2 = tf.placeholder(tf.string, shape=(None))

# For evaluation we use exactly normalized rather than
# approximately normalized.
sts_encode1 = tf.nn.l2_normalize(embed(sts_input1), axis=1)
sts_encode2 = tf.nn.l2_normalize(embed(sts_input2), axis=1)

# Get cosine similarity for comparison
cosine_similarities = tf.reduce_sum(tf.multiply(sts_encode1, sts_e
clip_cosine_similarities = tf.clip_by_value(cosine_similarities, 0
sim_scores = 1.0 - tf.divide(tf.acos(clip_cosine_similarities), 3.
```

The `placeholder` in the above code is just a variable which will be assigned data at a later date. This will let us pass in the sentence pairs when we run the TensorFlow session. We pass data to our placeholder using the `feed_dict` argument. This is what you can use to inject into a TensorFlow graph.

```
    """Returns the similarity scores"""
    emba, embb, scores = session.run(
        [sts_encode1, sts_encode2, sim_scores],
        feed_dict={
            sts_input1: questions['new_query'].tolist(),
            sts_input2: questions['question2'].tolist()
        })
    return (emba, embb, scores)
```

Don't worry if some of the TensorFlow concepts do not make much sense when you see them here. TensorFlow can take some getting used to since it does not run like a sequential program and instead you build up a dataflow graph where the nodes are mathematical operations and the connecting edges. For our example here you can look through the code to get a better idea for how it works. The rest of our code is just creating a dataframe to store the info we need and sort it so that we end up with a data frame which finds the top 5 similar sentences to our input query.

To check what sentences would be recommended as being similar to a potential new query simply run the program with a test sentence in a terminal, e.g.

```
python quora_recomend.py -q 'I lost my password, how can I get a n
```

Alternatively, you can run the '***quora_recomend.py***' as a notebook via the '***Quora Recommender***' notebook. It may be easier to follow each step in that way and enter your own test sentences to see how well they match the relevant answer groups

Run on FloydHub

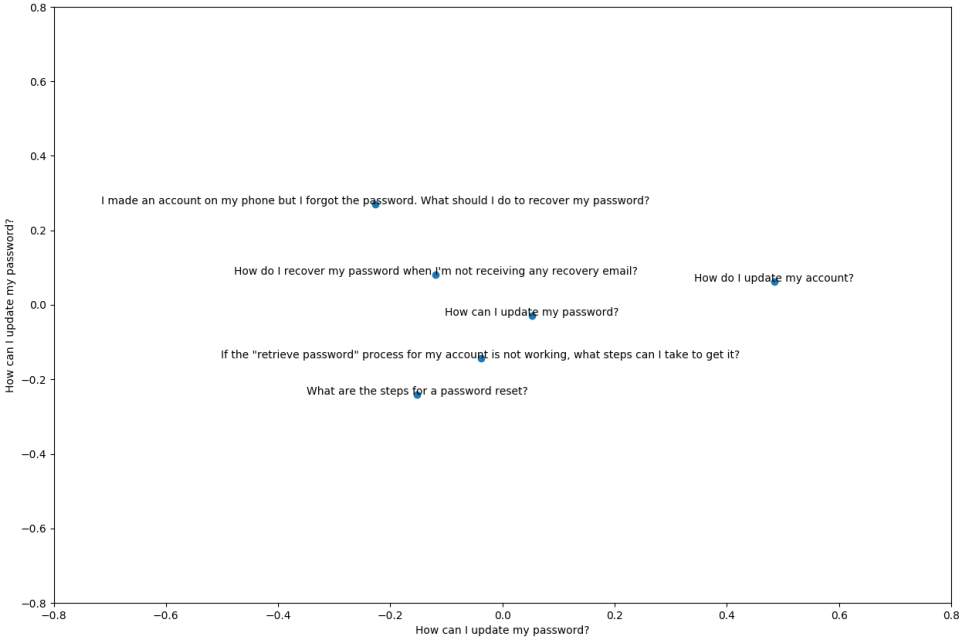
This program will:

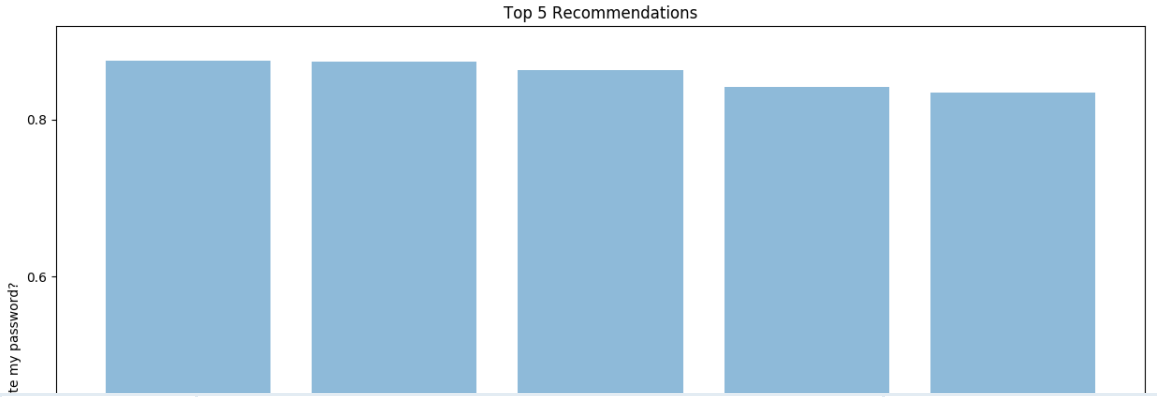
1. Generate an embedding for the submitted query and also for each saved query
2. Find the cosine similarity between the new query and each saved query embedding
3. Sorts the results based on the best matches, returning the top 5 best matches and the corresponding answer label

To visualize this the program also shows the embeddings via a 2D diagram. This uses the PCA dimension reduction technique we used in part 1 of the blog post to show how we could reduce a 4D vector to a 2D vector.

> Here we are reducing the dimensions from 512 to 2 to show which embeddings are closer.

The closer the embeddings the more likely they will return a high cosine similarity. The second graph is a bar chart showing the results from the cosine similarity scores.


I made an account on my phone but I forgot the password. What should I do to recover my password?
How do I recover my password when I'm not receiving any recovery email?
How do I update my account?
How can I update my password?
If the "retrieve password" process for my account is not working, what steps can I take to get it?
What are the steps for a password reset?
How can I update my password on?
How can I update my password?




| SIM_SCORE | NEW_QUERY | QUERY |
|---|---|---|
| 0.8748 | I lost my password, how can I get a new one | How do I recover |
| 0.8674 | I lost my password, how can I get a new one | I made an accour |
| 0.8573 | I lost my password, how can I get a new one | If the "retrieve pa |
| 0.8447 | I lost my password, how can I get a new one | I forgot my usern |
| 0.8430 | I lost my password, how can I get a new one | What are the step |

3: What are the steps for a password reset?

You can try this yourself with some test sentences to see if you can "beat" the system. Try and create a sentence that can "trick" the system to recommend an obviously incorrect answer.

4: I made an account on my phone but I forgot the password. What should I do to recover my password?

5: How do I update my account?

These are the queries which the system identifies as most similar. Note that the results

For example, let's ask the system: `How can i reset my cat?` oup.

> python quora_recomend.py -q 'How can i reset my cat?'

| SIM_SCORE | NEW_QUERY | QUERY |
|---|---|---|
| 0.7080 | How can i reset my cat? | How do you disable an account? |
| 0.7052 | How can i reset my cat? | What are the steps for a password res |
| 0.7018 | How can i reset my cat? | How do you delete an account? |
| 0.6937 | How can i reset my cat? | I made an account on my phone but I |

So from this table we can see that if we add a random word such as `cat` we still get some relatively high result such as 0.7080 for a "Account Delete" group. There is some similarity here with the reset word but overall we would like a lower scoring than 0.70. If we had a lot of questions it might be difficult to make differentiation between two different groups of questions with the word password it in for example.

We can certainly debate the relative similarity of the lower recommendations here. There is something there but it seems hard to believe that they are similar enough to represent a similar customer query. Our original sentence was about resetting a password but the second one was a semantically unrelated question.

We might not be confident to set this up to automatically respond to a customer, for example. Instead, the recommendation system works like a filter and provides a human agent with a reduced set of potential matches. They could then identify situations, like the cat reset, where the recommendations were not relevant and require the customer service agent to handle the fresh new request.

# Next steps and further research

Sentence embedding is a relatively new deep learning NLP method. It is still an area of active research. It's always interesting to look at these early stages of a new technology, but, ultimately, to get value from it, we want to know if we can use it now.

From our brief overview, it's clear there is much potential here. We have not tried any domain specific fine tuning nor have we tried to

find a better scoring mechanism than cosine similarity. As a result there is much to investigate!

You're probably wondering if you can use sentence embeddings in production right now. And the answer is yes – as outlined here, they could be used to reduce the workload on internal teams to find related questions to common customer queries.

Sentence embeddings seem well suited to providing support options to human operators to augment their decision making and not replace it. But good luck trying to figure out how to reset your customer's cat.

---

## About Cathal Horan

This is part two of a two-part blog series by Cathal Horan on sentence embeddings. You can read the first post here.

Cathal is interested in the intersection of philosophy and technology. Specifically how technologies such as deep learning can help augment and improve human decision making. He recently completed an MSc in Business Analytics. His primary degree is in electrical and electronic engineering. He also has a degree in philosophy and a MPhil in psychoanalytic studies. He currently works at Intercom. Cathal is also a FloydHub AI Writer.

You can follow along with Cathal on Twitter, and also on the Intercom blog.