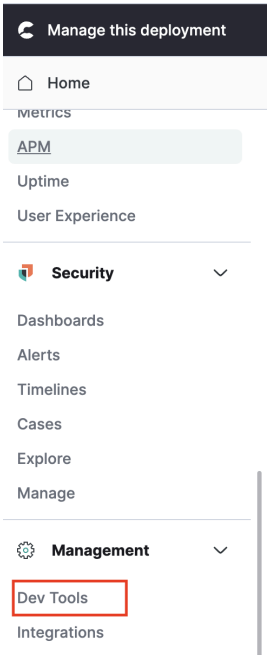
NLP Workshop Lab book

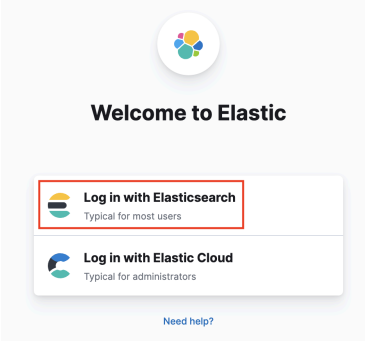
Getting Started



You'll be performing the hands-on work in a shared Elastic cluster. Below is how you can access that cluster:

URL: https://ela.st/nlp-workshop-cluster-emea
or https://ela.st/nlp-workshop-cluster2-emea
And "Log in with Elasticsearch" using these credentials:

User: guest
Password: nlp-workshop



In the more technical parts of the workshop, you'll interact with the data in Elastic using the developer console. If you haven't before, you can access it under "Dev Tools", as shown on the left (near the bottom of the burger menu).

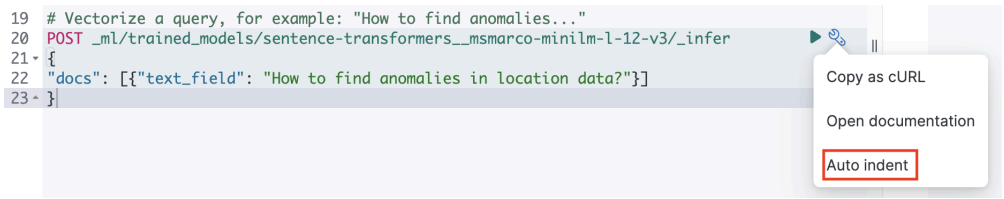
Scoring Tool

Throughout the workshop, you'll be challenged to provide solutions to some questions, which relate directly to the exercises you'll be performing.

We'll run a leaderboard during the workshop. Whenever you encounter a Challenge, provide your answer to the scoring tool, find the corresponding question on the "Challenges" tab. You'll need to register yourself at the beginning: https://elastic-vectorsearch.ctfd.io/

Lab 1: Semantic Search - Knowledge Base

In this lab, you will mine a database of knowledge (blog posts, specifically) for insight - finding posts that are similar to a query, filtering the semantic search with metadata, and obtaining answers to specific questions.
Throughout the lab, you'll copy Elastic code into Dev Tools. To preserve the required indentation, you can use "Auto Indent" (click on wrench icon as shown below), or type Ctrl-I, when you have the cursor at the active query.



Semantically Search Text

Semantic search overcomes some issues with keyword-based search since it captures the meaning of your query at a deeper level than specific keywords.

Using the index with embeddings for the blogs that we preloaded into the cluster, you can **search "semantically"**, for close matches to queries, like "How to find anomalies in location data?".

```
POST blogs-processed/_search
{
  "_source": ["body_content_window", "byline", "url"],
  "knn": {
    "field": "text_embedding.predicted_value",
    "k": 5,
    "num_candidates": 10,
    "query_vector_builder": {
      "text_embedding": {
        "model_id": "sentence-transformers__msmarco-minilm-l-12-v3",
        "model_text": "How to find anomalies in location data?"
      }
    }
  }
}
```

Challenge 1: "Create Anomaly Detection Workflow" *(10 points)*
What are the first two words identifying what you need to click to create an anomaly detection job?
Hint: No new query needed, just sift through the output from the previous query on "How to find anomalies in location data?"

Since semantic search got collapsed into a single API call, you lose sight of what's happening behind the scenes. As explained in the introduction, any vector search proceeds in the following two steps, fundamentally:
Step 1: converts your query into its numeric ("vector") representation in embedding space, and second, finds the nearest neighbors to the query among the indexed documents by performing a kNN search.
To illustrate what a vector embedding looks like, the code below "vectorizes" our query for finding anomalies:

```
# Vectorize a query, for example: "How to find anomalies..."
POST _ml/trained_models/sentence-transformers__msmarco-minilm-l-12-v3/_infer
{
  "docs": [{"text_field": "How to find anomalies in data?"}]
}
```

Challenge 2 "Get the Text Embedding" *(5 points)*
What's the first non-zero digit of the embedding of the query above?

Step 2: runs k-nearest neighbor search

```
# Run kNN search against [#vector] embedding obtained above
POST blogs-with-embeddings/_search
{
   "_source": ["body_content_window", "byline", "url"],
   "knn": {
      "field": "text_embedding.predicted_value",
      "k": 5,
      "num_candidates": 10,
      "query_vector": [#vector#]
   }
}
```

Please note: Don't forget to replace "#vector#" with the embeddings from the previous command.

Using Filters with kNN search

Semantic search with filter: Let's modify the query above on how to find anomalies, and filter on the author, which is stored in the "byline.keyword" field. We want to get only blogs authored by "Melissa Alvarez"

```
# Search text embedding with a filter
POST blogs-processed/_search
{
  "_source": ["url", "byline"],
  "knn": {
    "field": "text_embedding.predicted_value",
    "k": 5,
    "num_candidates": 10,
    "query_vector_builder": {
      "text_embedding": {
        "model_id": "sentence-transformers__msmarco-minilm-l-12-v3",
        "model_text": "How to find anomalies in data"
      }
    },
    "filter": {
      "term": {
        "byline.keyword": "Melissa Alvarez"
      }
    }
 }
}
```

Challenge 3: "Filter Similarity Query Results" *(10 points)*
Which month was the best matching blog authored by Melissa Alvarez published?

Note: You can find more information, including how num_candidates influences the search, on this doc page. That won't matter on this small dataset of just ~300 blog posts, but may be relevant for your own, larger data sets.

Question Answering

A “question answering” query mines a specific document (or set of documents) for an answer to a question, which is formulated in natural language. To try this out, go back to the first search above, and copy the "body_content_window" value from the _knn_search result, replacing TEXT with the whole `body_content_window` (from “About the …” all the way to the end) in the query below:

```
#  Run question & answer query
POST _ml/trained_models/deepset__tinyroberta-squad2/_infer
{
  "docs":[{"text_field": "TEXT" }],
   "inference_config": {
       "question_answering": {
             "question": "where did we get the data from?",
             "max_answer_length": 30
     }
  }
}
```

Challenge 4: “Answer Question with NLP” *(5 points)*
Which city did we get the data from?

If you want to tinker some more, you may also try to get the answer to “what do we need to click on?”

Sparse Semantic Search

Since 8.8 Elastic supports an overall yet simpler way to implement semantic search, with the Elastic Learned Sparse Encoder that provides performant semantic search on specialized domains without the need to tune an embedding model!

Since in this lab, we preconfigured the cluster with the appropriately processed indices for you, the simplification isn't going to be obvious to you - in both cases we processed the blogs data with an ingest pipeline:

- For ("dense") vector search we added (dense) vector embeddings like you just did in challenge 2, and stored in the field `text_embedding.predicted_value`.
- For "sparse retrieval" shown below we expanded the blog content by applying ELSER during ingestion, stored in the field `body_expanded.predicted_value`

```
POST blogs-processed/_search
{
  "_source": ["body_content_window", "byline", "url", "body_expanded.predicted_value"],
  "query":{
      "text_expansion":{
         "body_expanded.predicted_value":{
            "model_id":".elser_model_1",
            "model_text":"How to find anomalies in location data?"
         }
      }
  }
}
```

In the result, the sparse array for this document is visible under the field "body_expanded", underneath the "body_content_window".

Lab 2: Image Similarity - Ecommerce search

In this lab, you will learn how to conduct semantic searches on image data. We'll use data from a fictitious E-Commerce site called Gallivant and search for images of the product we're interested in, based on its textual description.

The original data is available on Hugging Face; to avoid execution delays during the workshop we selected a subset of <2,000 entries for this workshop.

Embeddings for the product images were created prior to the workshop - using the OpenAI CLIP model - and the image embeddings obtained were preloaded into your workshop cluster.

Since the image embeddings are 512 dimensional, we can't use the 384 dimensional MSMARCO model we used in Lab 1 to vectorize text queries against the blog data. Instead, we'll use a different text embedding model that's compatible with the image embeddings, whose model ID is `sentence-transformers__clip-vit-b-32-multilingual-v1`.

Textually Search Product Descriptions

Without vector search, Ecommerce sites typically search textual descriptions to find matching products.

Challenge 5: "Limitations of keyword-based search" *(5 points)*
What are reasons traditional, keyword-based search can yield irrelevant results: metadata missing or inaccurate, user not familiar with the terminology, all of the above?

Image Similarity Query
You may guess it - once you have represented your data in (dense vector) embeddings, running semantic searches works conceptually the same regardless of the type of original data.
To make that explicit, you can run similarity searches by converting either textual descriptions or images into their dense vector representation, and then pull similar entries from your data using nearest neighbor search.

1. Convert the textual description "blue t-shirt" into an embedding by running an inference (`_infer` endpoint) with the CLIP sentence transformer that generates 512 dimensional embeddings, matching the image embeddings.

```
#  Vectorize your query
POST _ml/trained_models/sentence-transformers__clip-vit-b-32-multilingual-v1/_infer
{
  "docs": [{"text_field": "blue T-shirt"}]
}
```

2. Run a knn search against the *ecommerce* index. Be sure to copy the embedding #vector you obtained in step 1 as `query_vector`, like you did during the semantic search on blogs in Lab 1. And indicate you want the *image_name*

```
#  Run kNN search against [#vector] embedding obtained above
POST ecommerce/_search
{
  "_source": ["image_name", "name"],
  "knn": {
    "field": "image_embedding.predicted_value",
    "k": 5,
    "num_candidates": 10,
    "query_vector": [#vector#]
  }
}
```

Challenge 6: "Image search" *(5 points)*
Based on the above image similarity search for blue t-shirts, which hue of blue is the best match?

Image and Text Search App

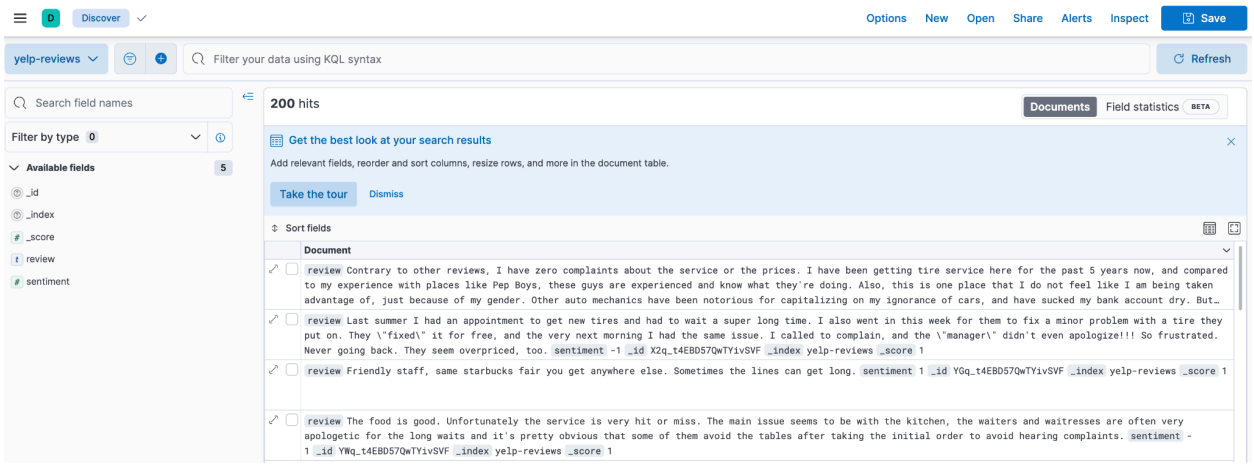
To solve this challenge, use one of the following links to the application:
https://ela.st/nlp-workshop-app-emea or https://ela.st/nlp-workshop-app2-emea

1. Navigate to the "Image Search" tab
2. Search for blue t-shirts

Challenge 7: "Product similarity search"
Are all returned t-shirts actually blue? (5 points)

Lab 3: Classifying user content

In this lab, you will apply classification to identify sentiment from text data, and assign other categories to it. We'll use a subset of Yelp reviews - the original data set is available on Hugging Face (Yelp reviews). You can explore the data in the Discover View of Kibana.



The key fields of this data are

- `review`: text field containing the actual review, which we abbreviated to 512 words
- `sentiment`: actual, the "ground truth" for sentiment analysis

Note: The 512 cap on tokens count comes from the BERT models which was trained that way.

Validate the NLP models

This lab uses three NLP models:

1. Assignment of sentiment: distilbert-base-uncased-finetuned-sst-2-english
2. Named Entity recognition: dslim__bert-base-ner
3. Zero-shot classification: distilbert-base-uncased-mnli

Let's first try out the sentiment model:

```
#  Obtain sentiment classification from some arbitrary sentence
POST _ml/trained_models/distilbert-base-uncased-finetuned-sst-2-english/_infer
{
  "docs": [
    {
      "text_field": "That plug really didn't work for me, I had to order a
different one"
    }
  ]
}
```

1. Classify Sentiment of Text

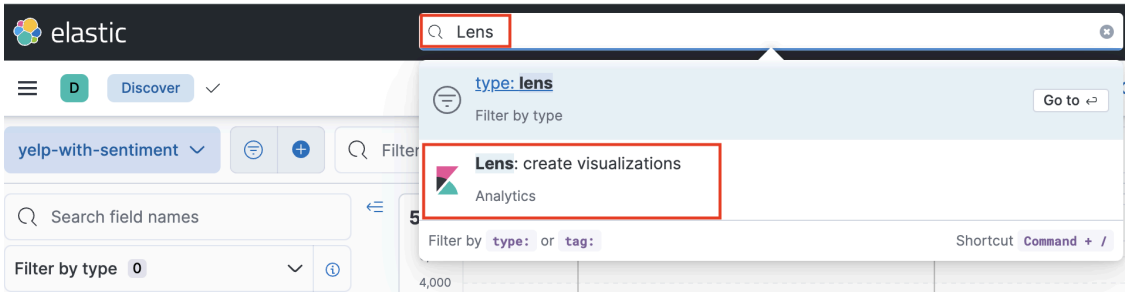
In this exercise, you'll apply a sentiment analysis to the text of these yelp reviews.

Attaching sentiments to data proceeds similarly to assigning (dense vector) embeddings: you need to define an inference pipeline (which in this case uses a sentiment model), and then applying the NLP model in an inference step, called re-indexing. To make this lab easier, we've already performed the reindexing and created an index *yelp-reviews-enhanced*

You can answer the first questions by inspecting the index on Kibana's Discover screen.

1. Challenge 8: Are the reviews balanced or primarily negative/positive? *(5 points)*

   Hints: To answer this question, create a visualization in Kibana on the sentiment category that was assigned to a field within sentiment-ml. Open the "Visualize Library" in Kibana, create a "Lens" visualization, drop sentiment-ml.predicted_value into the lens (on the canvas). - A little more direct, you can access the "Lens" visualization by typing it into directly the Kibana search bar:



2. Challenge 9: What percentage does the model correctly classify as "positive" or "negative"? *(10 points)*

   Hint: Either inspect the "Predicted vs Actual Sentiment" chart on the "Yelp reviews analysis" dashboard, or alternatively, create a visualization in Kibana comparing the predicted values of sentiment-ml with the annotated sentiment. You can use a table, heatmap, or donut sliced by actual and predicted sentiment.

2. Apply Named Entity Recognition

In this exercise, you'll extract which words represent entities like people, locations. That helps extracting the values of those entities for further processing by your application, or the additional abstraction can facilitate building advanced models, like predicting your affinity to various outdoor activities based on where you grew up.

```
POST _ml/trained_models/dslim__bert-base-ner/_infer
{
  "docs": [
    {
      "text_field": "Hi my name is <name> and I grew up in <town, state>"
    }
  ]
}
```

Note: replace the fields between <> with some fictional data of your choosing.

3. Classify without labeled data

In this final exercise, you'll apply the yelp reviews for categories that you can define on the fly by applying "zero shot classification", without having to curate labeled data and train an actual classifier.

The simplest way to demonstrate zero-shot classification is defining a couple categories and apply them directly to the text of a document as shown below.

```
# Directly apply zero-shot classification
POST _ml/trained_models/typeform__distilbert-base-uncased-mnli/_infer
{
  "docs":[{ "text_field": "I've always liked the auto bell and I go to several
around town. You can get out of the car while they clean it, or you can sit in your
car thru the wash and just drive off w/o waiting to get it dried, if you're in a
hurry. I've had issues with this particular auto bell recently, though. On at least
four occasions I've asked for the trunk of my car to be vacuumed only to find out
later it wasn't done. "}],
  "inference_config": {
          "zero_shot_classification": {
          "labels": [ "restaurant", "retail", "service", "entertainment", "hotel" ],
            "multi_label": true
          }
  }
}
```

Challenge 10: What type of business does the model classify Auto Bell? *(5 points)*

Resources

Below we are giving some suggestions on how you can learn more about Elastic and applying vector search.

- Frontend of the application
    - https://ela.st/nlp-workshop-app-emea
      https://ela.st/nlp-workshop-app2-emea
- Elastic 101 - if you're totally new to Elastic: ela.st/workshop-1-repo
- Technical blog on "traditional" BM25 scoring
- Install the Eland library and learn how to use it: https://github.com/elastic/eland, or blog describing all the ways you can access ML models in Elastic
- Learn more about Vector Search and NLP
    - What-is vector search page, provides high-level overview and links as below
    - Vector Search and NLP Webinar
    - Dense vectors, used to store embeddings: Doc page
    - (Approximate) Nearest Neighbors: Technical Blog, Doc page
    - Applying NLP models in Elastic: Doc page, Intro Blog, 3-Part Series, NLP in Elastic Tutorial (Community team on dev.to)
    - Create index pipelines with inference: "3. Creating a pipeline" and "4. Reindex the data" of the NLP blog on text embeddings and vector search
    - Ben Trent's answer about docs for question answering on discuss.elastic