PGT CS Programme-level Worksheet: Plagiarism and Academic Misconduct

Oxford University defines plagiarism as, ”presenting someone else’s work or ideas as your own, with or without their consent, by incorporating it into your work without full acknowledgement.” This is taken extremely seriously, and we want you to be able to avoid it.

The long and short of it is: **attribute (cite) all material that isn’t yours**. This includes:

- Directly quoted text (don’t forget to surround it with quotation marks!)
- Paraphrased or summarised text.
- Tables, figures, illustrations, photos, captions, and code.

This worksheet won’t delve into specific reference or bibliography formats - as long as you are consistent they don’t matter very much. Instead, it will cover some common pitfalls.

The university of Bristol Library guide to referencing has the following advice about Quoting, Paraphrasing, and summarizing:

Quoting

A word-for-word rewriting of a source's ideas.

When?

- Effective language that cannot be improved
- Direct support for a case you're making
- Expert declaration
- Controversial statement (to distance yourself!)

How?

- Verbatim
- Enclose in double quotation marks
- Indent for longer quotations
- Insert an ellipsis (...) if omitting words
- Add citation

Paraphrasing

A restatement or rewriting of a source's ideas of similar length using your own language whilst reflecting the source accurately.

When?

- Rearranging for emphasis
- Simplifying language
- Clarifying meaning
- To fit in with your own style
- To keep the same length

How?

- Read the source several times
- Write an outline in your own words
- Rearrange the outline for emphasis/clarity
- Write the paraphrase (without looking at the original!)
- Check against the original
- Add citation

Summarising

A shorter restatement or rewriting of a source's ideas using your own language whilst reflecting the source accurately.

When?

- To simplify
- To eliminate extra information
- To make a minor point
- To convey an overall meaning

How?

- Read the source several times
- Write an outline in your own words
- Rearrange the outline to suit your needs (change of emphasis?)
- Write the summary (without looking at the original!)
- Check against the original
- Add citation

1. Consider the following quote from Ishii and Ullmer[1]. It is correctly cited as a direct quotation. Try paraphrasing and then summarising it.

   ”The ambientROOM complements the graphically-intensive, cognitively-foreground interactions of the metaDESK by using ambient media – ambient light, shadow, sound, airflow, water flow – as a means for communicating information at the periphery of human perception” [1]

2. Below are several examples from TurnItIn reports. For each of them, consider and discuss what the student has done wrong and what they could do better to prevent bad academic practice and/or plagiarism.

[1]Hiroshi Ishii and Brygg Ullmer. 1997. Tangible bits: towards seamless interfaces between people, bits and atoms. In Proceedings of the ACM SIGCHI Conference on Human factors in computing systems (CHI ’97). Association for Computing Machinery, New York, NY, USA, 234–241. DOI:https://doi.org/10.1145/258549.258715

Abstract

With the rapid growth of information on the Internet, recommendation become more and more fundamental for liberating users from vast quantities of information and helping users find their target precisely. Recommender systems constitute the core engine of nowadays popular social network platform like twitter and facebook. The recent success of neural networks has been attracting more and more researchers to adopt neural network in recommender system.

Take Twitter as an example, despite the fact that Twitter data has been extensively used to understand socioeconomic and political phenomena and user behaviour, the implicit feedback provided by users on Tweets through their engagements on the Home Timeline has only been explored to a limited extent. At the same time, there is a lack of large-scale public social network datasets that would enable the scientific community to both benchmark and build more powerful and comprehensive models that tailor content to user interests. By releasing an original dataset of 160 million Tweets along with engagement information.

For large social network company like Twitter, the CTR (click through rate) of tweets on users' Home Timeline is quite essential. The accurate prediction of tweets users are more likely to engage in would increase user satisfaction and greatly benefits the company.

However, the feedback information we can get from user is implicit. They would not rate tweets which they would most likely to engage. This makes the recommendation task even harder. To tackle this complex, this project takes advantages of the hidden information in social network and designate a recommender system based on Node2Vec and FFM.

3.3 Listwise approach

The idea of listwise ranking objectives is to construct loss functions that directly maximize the effectiveness of the list. Instead of comparing two documents each time, listwise loss functions compute ranking loss with each query and their candidate document list together. The property of ranked list, documents are ordered by score, is maintained and ranking evaluation measure can be more directly incorporated into the loss functions in learning.

There are two main sub-techniques for listwise learning to rank: direct optimization of measures of information retrieval such as SoftRank [69] and AdaRank [73], and minimizing a loss function that is defined based on understanding the unique properties of the ranking such as ListNet [11] and ListMLE [39].

ListNet computes the probability distribution over all possible permutations based on model score and ground truth labels. The loss is then given by the KL divergence between these two distributions. ListMLE utilizes the likelihood loss of the probability distribution based on Plackett-Luce model for optimization.

Widely used measures in information retrieval such as NDCG and MAP obviously differ from the loss functions described above sections. In such a situation, a natural question to ask is whether the minimization of the loss functions can lead to the optimization of the ranking measures.

Researchers have attempted to answer this question, and it has been proved that the regression and classification based losses used in the pointwise approach are upper bounds of (1-NDCG) [41, 12, 5].

Though it is known that listwise approaches are more desirable for learning to rank because of the nature of the design, as shown in [6], pointwise and pairwise approaches tend to be preferred for performance reasons in practice [79].

2.4.2 User Temporal Analysis

Using events chosen based on the forum temporal analysis, enables us to perform the analysis of individual users. The goal is to take a closer look at individual sentiment scores and how they would change on a day-to-day basis during the month of a major terrorist event.

Initially, a top 20 list was created for the most negative, and positive users based on their average sentiment scores over their forum posting lifetime. This was done to compare the difference between our negative users and positive users. Any users with less than 500 posts were eliminated from the running due to limited forum use. This provided us with a base to start individual analysis around terrorist events. However, we soon realized that only 4-5 users from the top 20 negative posters and 1-5 users from the top 20 positive users were active during the month of each event. Thus top 10 lists for positive and negative users were drafted for each event based on the average sentiment scores of users only during that month, ensuring that all 10 users are active during the month of the event.[1]

```
  def call(self, x, training):
    # x shape == (bs, 256, 256, 3)
    x1 = self.down1(x, training=training) # (bs, 128, 128, 64)
    x2 = self.down2(x1, training=training) # (bs, 64, 64, 128)
    x3 = self.down3(x2, training=training) # (bs, 32, 32, 256)
    x4 = self.down4(x3, training=training) # (bs, 16, 16, 512)
    x5 = self.down5(x4, training=training) # (bs, 8, 8, 512)
    x6 = self.down6(x5, training=training) # (bs, 4, 4, 512)
    x7 = self.down7(x6, training=training) # (bs, 2, 2, 512)
    x8 = self.down8(x7, training=training) # (bs, 1, 1, 512)

    x9 = self.up1(x8, x7, training=training) # (bs, 2, 2, 1024)
    x10 = self.up2(x9, x6, training=training) # (bs, 4, 4, 1024)
    x11 = self.up3(x10, x5, training=training) # (bs, 8, 8, 1024)
    x12 = self.up4(x11, x4, training=training) # (bs, 16, 16, 1024)
    x13 = self.up5(x12, x3, training=training) # (bs, 32, 32, 512)
    x14 = self.up6(x13, x2, training=training) # (bs, 64, 64, 256)
    x15 = self.up7(x14, x1, training=training) # (bs, 128, 128, 128)

    x16 = self.last(x15) # (bs, 256, 256, 3)
    x16 = tf.nn.tanh(x16)

    return x16
class DiscDownsample(tf.keras.Model):
  def __init__(self, filters, size, apply_batchnorm=True):
    super(DiscDownsample, self).__init__()
    self.apply_batchnorm = apply_batchnorm
    initializer = tf.random_normal_initializer(0., 0.02)

    self.conv1 = tf.keras.layers.Conv2D(filters,
                                        (size, size),
                                        strides=2,
                                        padding='same',
                                        kernel_initializer=initializer,
                                        use_bias=False)
    if self.apply_batchnorm:
      self.batchnorm = tf.keras.layers.BatchNormalization()

  def call(self, x, training):
    x = self.conv1(x)
    if self.apply_batchnorm:
      x = self.batchnorm(x, training=training)
    x = tf.nn.leaky_relu(x)
    return x

class Discriminator(tf.keras.Model):

  def __init__(self):
    super(Discriminator, self).__init__()
```