Project Checkpoint #1

Kristen Vinh
kv1895@rit.edu

I. RECENT PROGRESS

The first few weeks of development on the "Applications of Machine Learning for Knitting Sweater Pattern Recommendation" project have focused on data acquisition and processing to ensure a robust dataset comprising sweaters that vary in attributes. I also began testing the first machine learning algorithm.

The acquisition process entailed:

- A three-part process to download knitting patterns from the Ravelry API and all their related details:
    1. An initial search using the "search.json" function of the API to search for patterns based on specific criteria. After much trial and error (searching just "sweater" vs. "cardigan", adding knitting to the search term, and identifying characteristics that could easily (craft type = knitting) or only be obtained via search (number of colors), the following acquisition criteria were developed:
        - searched by sweater type ("pullover" and "cardigan") and craft ("knitting") and number of colors (1, 2, 3, 4, 5, more than 6)
        - Page through at least 20,000 patterns
        - Combine the lists, filter out duplicates
    2. A retrieval of all the pattern details using the pattern's "ID" field (using "/patterns/{pattern_id}.json") to return further needed details, such as:
        - attributes
        - gauge
        - gauge pattern
        - yarn weight
    3. A script to download the medium-sized image for each of the sweaters. Many of these failed, and an additional script was written to exclude patterns for which no photo could be downloaded.

- Downloading all available attributes for all patterns in Ravelry from "/pattern_attributes/groups.json", which was initially cleaned to remove attributes that did not apply to knitted sweaters (i.e., sock pattern attributes, crochet attributes) or applied to the pattern type (i.e., written pattern, charts). It was then further cleaned to remove invisible attributes (such as illusion colorwork and sweater ease) as well as attributes standard to all sweaters (hems, straight).
- Attributes were then organized in categories based on Ravelry's hierarchical system (colorwork, design elements, collar, neckline, sleeve, and other characteristics), with one additional category, "sweater construction," that consisted of tags across multiple categories. This was done because construction (yoke, raglan sleeve, drop sleeve) is often a key descriptor of what a finished knit sweater looks like. Additionally, I created two new tags for the colorwork category (attr_single_color and attr_mult_color) based on the searches, as the colorwork category did not previously distinguish between these.

At this point, I determined that focusing on five main categories for algorithm testing would be most beneficial: sweater type, sweater construction, colorwork, sleeves, and necklines. The other categories (design elements, other characteristics) will be tested if time permits.

Also, during this process, it was discovered that many sweaters are poorly tagged, and I began a process to continue to explore and clean the data:

- I conducted exploratory data analysis to plot the number of sweaters in each attribute I wanted to examine within each category.
- I also performed some data cleaning — if a sweater did not have characteristics in at least two out of the three categories (colorwork, sleeves, or neck), it was dropped from the database. The "sweater type" was not used as a criterion because all the samples were downloaded in a way that ensured they had a sweater type.
- I removed tags that would not be helpful in identification – such as "attr_sleeves", since this seemed to be poorly tagged. Most sweaters have sleeves (unless tagged as "attr_sleeveless"). I also removed attributes such as "attr_contiguous", a sleeve tag that indicates that the sleeves are knit directly from the body, but is not a characteristic that would be visible to a machine learning algorithm.
- One of the significant issues I'll need to address in the next step is the data imbalance of tags and how to handle tags with very few entries.

Examples of charts created during the exploratory data analysis portion (on cleaned samples) include Figure 1, which shows the distribution of cardigans versus pullovers in the dataset, and Figure 2, which displays the distribution of sleeve tags (used here to demonstrate that some tags have very few samples).

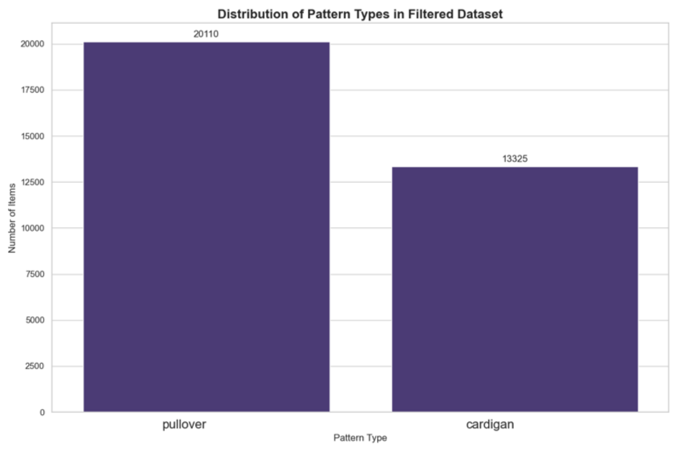

Fig 1. Distribution of pullover vs. cardigan

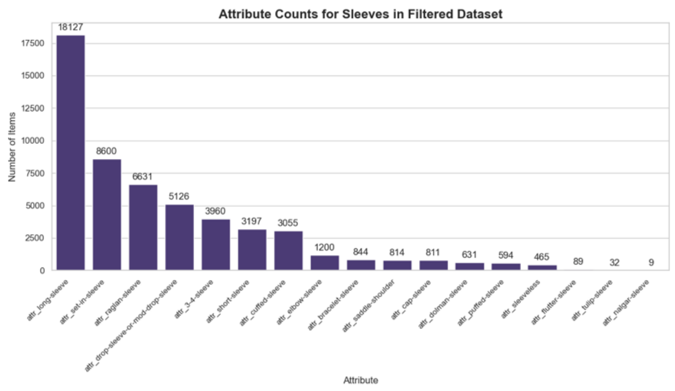

Figure 2. Sleeve tag distribution – demonstrates that some tags have very few samples.

This indicates that class imbalance is something I will need to address as I enter the second phase of my research project.

As the final part of this project phase, I wanted to test the most basic of the algorithms, LeNet-5, on the classification of cardigans versus pullover sweaters. This algorithm was not very successful, achieving a test accuracy of only 64%. The confusion matrix for the test data is shown in Figure 3.

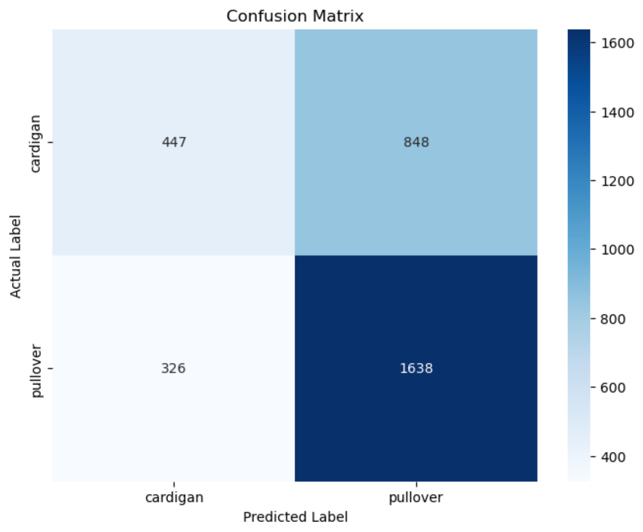

Fig 3. Confusion matrix for the LeNet5 model run on cardigan and pullover labels.

This suggests that I need a more robust, pre-trained model, such as VGG-19, as the model is making many incorrect classifications of cardigans. Tests I conducted while developing the proposal showed that more advanced algorithms (such as VGG-19) performed much better on this classification, even with smaller datasets. I've also identified one issue that may be contributing to cardigans being misclassified: photos taken from the back of the sweater, as shown in Figure 4.

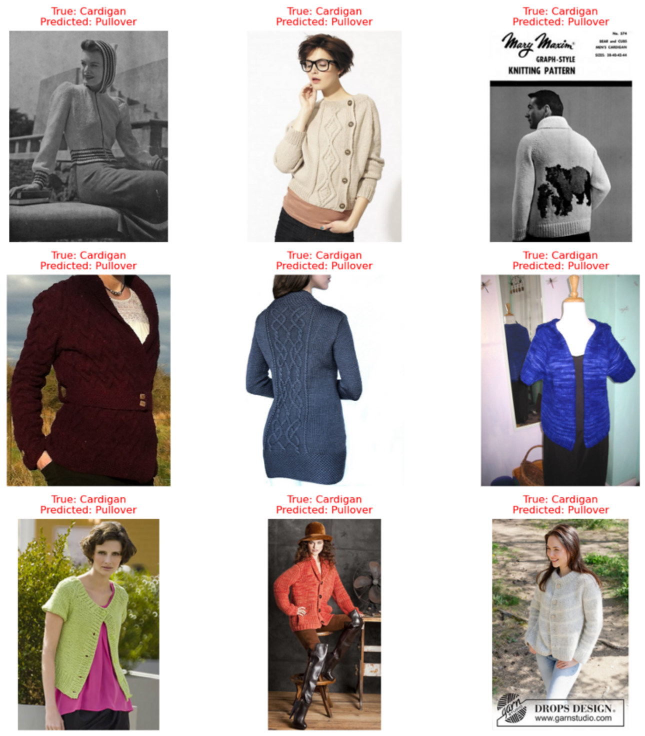

Fig 4. Incorrectly labeled cardigan images.

I'll need to determine if a more robust model can address misclassification issues like this one or if an alternative solution is necessary.

II. QUESTIONS/CONCERNS

- What is the best way to handle data that has been inconsistently tagged? Is the current approach of filtering the dataset only to include samples with tags in the categories I want to examine sufficient?
- As indicated above, some of the cardigans are being misclassified, possibly because they're photos of the back of the sweater. I'll examine whether a more robust model handles this or if another method needs to be researched, but I'm a little concerned about developing models that can handle complex tasks at this point.
- When building predictive models for construction type (for example), is it better to separate the dataset by type (e.g., cardigans vs. pullovers) and train a separate model for each, or to train a single model on the entire dataset? I believe that experimenting with both methods will be necessary to determine which one works best, but I'm curious to know if there are established best practices for this approach.
- What is the best way to handle attributes that have very few (fewer than 10) sweaters tagged? Can they be removed as classes?
- I'm below my goal for sweater counts of at least 15,000 for each sweater type (although not by a lot), but I think it's better to have well-tagged data versus having more poorly tagged data – is this the right approach?
- Is it acceptable to access the API and download additional data to enhance a tag? For instance, the "circular-yoke" category on Ravelry shows 9,000 sweaters, but my dataset has only a third of that.

III. NEXT STEPS

1. Begin the process of more in-depth testing. I will start by optimizing the easier-to-define categories (sweater type, sweater construction, and colorwork) before moving on to other characteristics for classification. Through my exploratory data analysis, I have identified that some categories require further breakdown for optimal classification (e.g., sleeve length: long, elbow, ¾ sleeve, and sleeve type: raglan, set-in, drop, etc.). I'll likely identify further breakdowns during this part of my testing.

   Algorithms and categories for testing:

   1. Algorithms:
      - i. LeNet5 (continue to test on other classifications beyond type)
      - ii. AlexNet
      - iii. ResNet50
      - iv. VGG-19
      - v. Annoy
   2. Categories:
      - i. Sweater Type
      - ii. Sweater Construction
      - iii. Colorwork
        1. Single Color vs. Multi-color
        2. Types of Colorwork
      - iv. Neckline
      - v. Sleeve
        1. Sleeve Length
        2. Sleeve Type
2. Download the real-life data from Reddit for testing.
3. Develop a system to benchmark the most effective algorithms, including using test samples and explainable AI.