**Introduction to Practical Artificial Intelligence**

**EXAM FS 2021**

A problem is given to a student for 2 hours. In this timeslot students should complete the work and upload it to Moodle. As soon as he/she is ready, the student defends the solution orally by showing the code and results. Professor can ask one additional question, which may require code update. Link: https://docs.google.com/document/d/e/2PACX-1vSovVDXlgwA0uO\_T2lKSmNfjAgEm94bbunQ1qrQ72d1mud3TRqMnPogy3jsHmMn90o2P9UhaNNzSXPH/pub

**Declarative AI**

**Problem 1. Symbolic computations**

1. [75%] Find local maxima (and only maxima!) (x and f(x)) for this function using symbolic computations library in Python (e.g. sympy):

f(x) = x7 - 10x6 + 42x5 + 128x4 - 800x3 + 1300x2 - 670x

1. [25%] Visualize the result as a graph, and show maxima points.

**Problem 2. Solving**

1. [75%] Implement text interface (bot), which can solve computational problems, equation solving and differential equation solving. The bot should return relevant answers with respect to the task (number for computations and equations, functions for differential equations).
2. [25%] type of: 34 \* 45  ——> 1530
3. [25%] type of: 2\*x^2 + 4x + 2 = 0 ——> -1
4. [25%] type of: f’ = cos(x) ——> f = sin(x) + C
5. [23%] When result is the function — plot the graph.
6. [2%] For equation solving (if possible) do the plot (intersection of 2 graphs)

**Problem 3. Minimax**

1. [75%] Implement a minimax tree for a simplified version of [Halma](https://www.google.com/url?q=https://en.wikipedia.org/wiki/Halma&sa=D&source=editors&ust=1632386317179000&usg=AOvVaw2q3FjY0OmNUpay4BbqmCTx). You are given 3x3 field and 3 pieces each:

|  |  |  |  |
| --- | --- | --- | --- |
|  | A | B | C |
| 1 | X | X |  |
| 2 | X |  | O |
| 3 |  | O | O |

You need to move pieces in the “Home” of your opponent (colored). Who finished first — wins. If you cannot finish in 16 moves — draw.

Moves are done into a spare neighbour cell (8-connectivity), OR by jumping over the piece (no matter, your or opponent’s). E.g. debuts like these are allowed:

* A2-B2 ; C2-A2
* A1-B2 ; C3-A1
* B1-C1 ; C3-B2

1. [25%] Visualize the game which leads to WIN. OR implement a bot to play with.

**Natural language processing**

**Problem 4. Syntax**

1. [75%] Create a language syntax (context-free grammar) which recognizes the following sentences, describing relations:

Moscow is far from Voronezh.

Kazan is far from Voronezh.

Kazan is far from Moscow.

Innopolis is close to Kazan.

Dolgoprudnyy is close to Moscow.

NizhniyNovgorod is equidistant to Moscow and Kazan.

Bor is close to NizhniyNovgorod.

Tula is equidistant to Moscow and Voronezh.

SaintPetersburg is far from Moscow.

Bologoe is equidistant to Moscow and SaintPetersburg.

Vyborg is close SaintPetersburg.

Zaymishche is equidistant to Innopolis and Kazan.

Let “far from” be 10, “close to” = 1, and “equidistant to” splits the existing edge into equal edges and transit vertex. Process sentences from the example and store information in the adjacency matrix.

1. [25%] Visualize the graph using networkx using spring layout. Is it similar to the real map?

**Problem 5. Syntax tree**

1. [75%] Alphabet is “0123456789()+-/\*”. Using this alphabet you can build arithmetic formulas with a mandatory brace. Like this:
2. (1)
3. (2+3)
4. ((2-4)\*(8/2))

        Write a syntax for this language. Restore AST.

1. [25%] Use the tree to compute the value of the formula.

**Audio processing**

**Problem 6. Music generation**

1. [75%] Generate and play a major chord for a given note (A = 440 Hz).
2. What is adding a semitone? Multiplying a note frequency by 21/12:   
                   440Hz + 1 semitone = 466.16Hz.
3. What is a major chord? It consists of 3 notes: a given one (frequency N), N+(4 semitones), N+(7 semitones).
4. [25%] Add a minor chord (N, N+3, N+7) implementation.
5. Play the melody “C G Am F G C”, where the capital letter corresponds to the note. Chord is major by default, “m” modifier means “minor” (Am = minor A chord). Note frequency [reference is here](https://www.google.com/url?q=https://pages.mtu.edu/~suits/notefreqs.html&sa=D&source=editors&ust=1632386317189000&usg=AOvVaw241-kPt6y92v9ZPkab7Yn-) (A=440, C=261.63, F=349.23, G=392.00). Each chord should last for 500ms.

**Problem 7. Echo**

1. [50%] Implement echo effect for a [given audio file](https://www.google.com/url?q=https://raw.githubusercontent.com/hsu-ai-course/hsu.ai/master/code/datasets/sound/peterpiper.wav&sa=D&source=editors&ust=1632386317192000&usg=AOvVaw1aquseW5-SrbK3o27KvxjC).
2. [50%] Implement two effect sliders (2 parameters) for both reflector distance and echo strength.

**Speech processing**

**Problem 8. AutoRap**

1. [75%] Choose a poem you like (or write a poem in English or Russian), and make your code read it with a musical or [beat](https://www.google.com/url?q=https://freemusicarchive.org/genre/Hip-Hop_Beats&sa=D&source=editors&ust=1632386317192000&usg=AOvVaw2AsnVTb-phTgrv18QTmCSV) background.
2. [25%] Align reading with a rhythm and volume of music/beat.

**Problem 9. Auto subtitles**

1. [80%] Generate subtitles for a video or an audio file [[e.g. this audio](https://www.google.com/url?q=http://sprotasov.ru/podcast/origin-12.mp3&sa=D&source=editors&ust=1632386317193000&usg=AOvVaw1PNBMnprizsUDXFQljpw5i) or [video](https://www.google.com/url?q=https://youtu.be/P5mlg91as1c&sa=D&source=editors&ust=1632386317193000&usg=AOvVaw1uIM5c0gVBFlTJftMJD1xP)] in [VTT format](https://www.google.com/url?q=https://en.wikipedia.org/wiki/WebVTT&sa=D&source=editors&ust=1632386317193000&usg=AOvVaw1eZqKZdhSGp6u2hDiO3aY8).
2. [20%] Estimate the quality of recognition using [WER](https://www.google.com/url?q=https://en.wikipedia.org/wiki/Word_error_rate&sa=D&source=editors&ust=1632386317194000&usg=AOvVaw1aUDPksy3tNA2lI3J3drbn).

**Machine learning**

**Problem 10. Classification**

1. [75%] For a [given image](https://www.google.com/url?q=https://github.com/hsu-ai-course/hsu.ai/blob/master/code/datasets/ml/PAI.png&sa=D&source=editors&ust=1632386317195000&usg=AOvVaw1KIBQdoLzk63l7Dtm1xopQ) recognize the letters. Use whatever approach you like. I can suggest [EMNIST](https://www.google.com/url?q=https://pypi.org/project/emnist/&sa=D&source=editors&ust=1632386317195000&usg=AOvVaw02OEDZ6RxJiYhkH8y6rEu3) or [pytesseract](https://www.google.com/url?q=https://pypi.org/project/pytesseract/&sa=D&source=editors&ust=1632386317195000&usg=AOvVaw2KFTaqORvxA-sHpUnkiDbN).
2. [25%] Build words out of the letters. Try to fix typos using [Norvig’s spell checker](https://www.google.com/url?q=https://norvig.com/spell-correct.html&sa=D&source=editors&ust=1632386317196000&usg=AOvVaw3xIx2l0i8355yV5V9H8pRU).

**Problem 11. Clustering**

1. [75%] For the [given list of topics](https://www.google.com/url?q=https://github.com/hsu-ai-course/hsu.ai/tree/master/homeworks/13&sa=D&source=editors&ust=1632386317196000&usg=AOvVaw0YDVBjY5NVFWgf7ddr3YAy), find their embedding and cluster them. Find the best number of clusters in the range of 3-20 based on the Silhouette score.
2. [25%] Find top-2 words, closest to the cluster centroid.

**Image and video processing**

**Problem 12. Tracking**

1. [75%] Track a cat on [this video](https://www.google.com/url?q=https://github.com/hsu-ai-course/hsu.ai/blob/master/labs/exam/cat.mp4&sa=D&source=editors&ust=1632386317197000&usg=AOvVaw0QEWrL2Oy5BP4U6rFRwpF7) (0:03 - 0:25) with a bounding box. Sample 3 frames per second.
2. [25%] Put a bounding box on all frames and generate a new video.

**Problem 13. Video tagging**

1. [75%] What is in [the video](https://www.google.com/url?q=https://github.com/hsu-ai-course/hsu.ai/blob/master/code/datasets/ml/cut.mp4&sa=D&source=editors&ust=1632386317198000&usg=AOvVaw34GJyhgei_RYM8FgnmAkb9)? Generate tags for video frames. Extract most confident tags.
2. [25%] Generate subtitles with tags in [VTT format](https://www.google.com/url?q=https://en.wikipedia.org/wiki/WebVTT&sa=D&source=editors&ust=1632386317198000&usg=AOvVaw1BUOX-6PGOl9blxnQ73E9W).

**ML infrastructure**

**Problem 14. Streamlit**

1. [100%] Create a [streamlit](https://www.google.com/url?q=https://www.streamlit.io/&sa=D&source=editors&ust=1632386317199000&usg=AOvVaw0JP_-oVc5KVgPuNflvjlXU) application, which will visualize DBScan algorithm behavior for 2D data.
2. [20%] Generate a 2D dataset using [make\_blobs](https://www.google.com/url?q=https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html%23sklearn.datasets.make_blobs&sa=D&source=editors&ust=1632386317199000&usg=AOvVaw0wG8ce6nIp7PEI-cvy1867).
3. [30%] Implement DBScan clustering for this data. Visualize with pyplot/seaborn/...
4. [50%] Wrap this implementation into streamlit application. minPts and eps should become sliders in interface.