1 Introduction

The goal of the project is to create a language proximity analysis system utilizing:

- Phonetic dictionaries (IPA) from WikiPron
- Word2Vec algorithm (CBOW) for learning vector representations
- Visualization tools for result analysis

2 System Architecture

The system consists of the following modules:

2.1 Data Extraction Module (`src/data/data_scraping`)

2.1.1 Phoneme Extraction

The `phoneme_extractor.py` script downloads phonetic dictionaries from WikiPron:

- Support for over 100 Latin-script languages
- Output format: word + IPA transcription (tab-separated)
- Filtering words containing digits and leading apostrophes
- Saving to `data/<lang>/phonemes.txt`
- Automatic skip if data already exists

2.2 Dataset Module (`src/data/datasets`)

2.2.1 MultilingualWordDataset

PyTorch dataset for character-level data:

```
class MultilingualWordDataset(Dataset):
    """
    Returns: (character, language_label)
    """
```

2.2.2 MultilingualPhonemeDataset

PyTorch dataset for phonetic data:

```
class MultilingualPhonemeDataset(Dataset):
    """
    Parses IPA strings into individual phonemes
    Returns: (phoneme, language_label)
    """
```

2.3 Word2Vec Module (`src/embeding`)

2.3.1 CBOW Model

Implementation of the Continuous Bag of Words algorithm:

$$\mathbf{h} = \sum_{-c \le j \le c, j \neq 0} \mathbf{v}_{w_{t+j}} \tag{1}$$

$$p(w_t | w_{t-c}, \ldots, w_{t+c}) = \text{softmax}(\mathbf{W}\mathbf{h}) \tag{2}$$

where:

- c – context window size
- $\mathbf{v}_{w_i}$ – context word embedding
- $\mathbf{h}$ – summed context vector
- $\mathbf{W}$ – output layer weight matrix

2.3.2 Network Architecture

- **Input layer**: Embedding ($|V| \times d$)
- **Hidden layer 1**: Sum of context embeddings
- **Hidden layer 2**: ReLU activation (128 units)
- **Output layer**: Linear ($128 \times |V|$) + LogSoftmax
- **Loss function**: Cross-entropy
- **Optimizer**: Adam

2.3.3 Hyperparameters Configuration

Training parameters are centralized in `src/embeding/hyperparamiters.py`:

- `LANGUAGES`: List of language codes
- `DATA_TYPE`: 'words' or 'phonemes'
- `EMBEDDING_DIM`: Embedding dimension (default: 100)
- `WINDOW_SIZE`: Context window size (default: 2)
- `EPOCHS`: Number of training epochs (default: 10)
- `BATCH_SIZE`: Batch size (default: 128)
- `LEARNING_RATE`: Learning rate (default: 0.001)

2.4 Logging System (src/logging)

Centralized logging configuration in `logging_config.py`:

- Separate log files per module in `logs/` directory
- Console output with INFO level
- File output with DEBUG level and timestamps
- Format: `YYYY-MM-DD HH:MM:SS - module - LEVEL - message`

2.5 Testing Module (tests/)

2.5.1 Dataset Verification

The `verify_datasets.py` script:

- Loads datasets and writes content to files
- Compares dataset output with original files
- Detects parsing errors (e.g., tab characters in word data)
- Validates both MultilingualWordDataset and MultilingualPhonemeDataset

2.6 Docker Support (docker/)

Containerization for reproducible environment:

- `Dockerfile`: Python 3.11 base image
- `docker-compose.yml`: Service definition with volume mounts
- `docker-init.sh`: Automated build and start script
- `.dockerignore`: Excludes cache and generated files

3 Experimental Results

3.1 Data Extraction

3.1.1 Statistics for Polish Language

Data type	Number of elements	Source
Unique words	26,258	Wikipedia (100 articles)
Characters	224,235	Flattened words
Phoneme entries	132,558	WikiPron
Phonemes	1,061,160	Flattened IPA

Table 1: Statistics of extracted data for Polish language

3.2 CBOW Training

3.2.1 Test on Simple Text

Text: *"the quick brown fox jumps over the lazy dog"*

Parameter	Value
Vocabulary size	27 characters
Training samples	39
Embedding dimension	10
Window size	2
Epochs	20
Initial loss	3.34
Final loss	2.22

Table 2: Training parameters on simple text

Example similarities:

- 't' ↔ 'e' (0.61)
- 'o' ↔ 'd' (0.53)
- 'i' ↔ 'o' (0.44)

3.2.2 Test on Polish Data

10,000 tokens from the Polish dataset:

Parameter	Value
Vocabulary size	38 characters
Training samples	9,996
Embedding dimension	50
Epochs	5
Initial loss	3.36
Final loss	2.61

Table 3: Training parameters on Polish data

Learned character similarities:

- 'i' ↔ 'o' (0.44)
- 'e' ↔ 'i' (0.38)
- 'a' ↔ 'e' (0.26)

4 Usage Instructions

4.1 Data Extraction

4.1.1 Single Language

```
# Phonemes
python src/data_scraping/phoneme_extractor.py pl
```

4.1.2 Multiple Languages (Runner)

```
python src/data/runner.py pl en de
```

4.2 Model Training

Training parameters are configured in `src/embeding/hyperparamiters.py`:

```
# Edit hyperparamiters.py
LANGUAGES = ['pl', 'en']
DATA_TYPE = 'phonemes'  # or 'words'
EPOCHS = 10
EMBEDDING_DIM = 100
```

Then run training:

```
python src/embeding/train_cbow.py
```

Models are saved to `models/` directory with timestamp.

4.3 Docker Usage

4.3.1 Build and Start Container

```
cd docker
./docker-init.sh
```

4.3.2 Access Container

```
docker-compose -f docker/docker-compose.yml exec \
    language-proximity /bin/bash
```

4.3.3 Run Training Inside Container

```
# Inside container
python src/embeding/train_cbow.py
```

5 Conclusions

5.1 Achievements

1. Implemented complete pipeline for data extraction from Wikipedia
2. Integrated WikiPron phonetic dictionaries (337 languages)
3. Created PyTorch datasets for multilingual data
4. Implemented CBOW algorithm from scratch
5. Created visualization tools (t-SNE, PCA, heatmaps)

5.2 Observations

- CBOW model effectively learns character representations
- Characters with similar usage have higher cosine similarity
- Training loss decreases steadily
- t-SNE/PCA visualizations show sensible structure in embedding space

5.3 Future Directions

1. Training on larger datasets (full Wikipedia dumps)
2. Implementation of Skip-gram as alternative to CBOW
3. Cross-lingual analysis (comparing embeddings of different languages)
4. Negative sampling for training acceleration
5. Hierarchical softmax
6. Visualization tools (t-SNE, PCA, similarity heatmaps)
7. Automated testing suite

6 References

1. Mikolov, T., et al. (2013). "Efficient Estimation of Word Representations in Vector Space." arXiv:1301.3781
2. Mikolov, T., et al. (2013). "Distributed Representations of Words and Phrases and their Compositionality." NIPS 2013
3. WikiPron: `https://github.com/CUNY-CL/wikipron`

A Project Structure

```
project/
|-- data/                    # Extracted data
|   `-- <lang>/
|       `-- phonemes.txt     # Word + IPA
|-- src/
|   |-- data/
|   |   |-- data_scraping/  # Extraction scripts
|   |   |-- datasets/       # PyTorch datasets
|   |   `-- runner.py       # Data pipeline
|   |-- embeding/
|   |   |-- cbow.py         # CBOW model
|   |   |-- train_cbow.py   # Training script
|   |   `-- hyperparamiters.py  # Config
|   `-- logging/
|       |-- logging_config.py   # Logger setup
|       `-- add_logging.py      # Helper script
|-- tests/
|   `-- verify_datasets.py  # Dataset tests
|-- docker/                  # Docker configuration
|   |-- Dockerfile
|   |-- docker-compose.yml
|   `-- docker-init.sh
|-- logs/                    # Log files
|-- models/                  # Trained models
`-- report/                  # Documentation
```

B Configuration Parameters

Parameter	Default value	Description
EMBEDDING_DIM	100	Embedding dimension
WINDOW_SIZE	2	Context window size
EPOCHS	10	Number of epochs
BATCH_SIZE	128	Batch size
LEARNING_RATE	0.001	Learning rate
MAX_TOKENS	None	Token limit (None = all)
DATA_TYPE	'phonemes'	'words' or 'phonemes'
LANGUAGES	['pl']	List of language codes

Table 4: CBOW training configuration parameters (`hyperparamiters.py`)