Using RSS Feeds to Support Second Language Acquisition
Roadmap

1. Introduction to Zeeguu
2. Demo
3. Architecture
4. Article Recommender
5. Conclusion
1. Introduction to Zeeguu

• Three fundamental principles
  • Only read the stuff you like
  • Have your words everywhere with you
  • Practice with personalized exercises

• Introducing Zeeguu Reader for Android
  • RSS Reader with Feedly synchronization
  • Learn anywhere while reading
  • Provides article recommendations
## 2. Demo

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>All articles</td>
<td>2797</td>
</tr>
<tr>
<td>Saved for later</td>
<td>0</td>
</tr>
<tr>
<td>Zeeguu Recommended</td>
<td>300</td>
</tr>
<tr>
<td>Nachrichten</td>
<td>2185</td>
</tr>
<tr>
<td>Tech</td>
<td>612</td>
</tr>
<tr>
<td>ComputerBase</td>
<td>142</td>
</tr>
<tr>
<td>Golem.de</td>
<td>224</td>
</tr>
<tr>
<td>heise online News</td>
<td>246</td>
</tr>
<tr>
<td>Fun</td>
<td>24</td>
</tr>
</tbody>
</table>
3. Architecture: User Interface

- **Activity**
  - Main application component
  - Provide the window for the user interface
  - Handle communication between fragments
  - This app: MainActivity, SettingsActivity

- **Fragment**
  - Reusable portion of user interface
  - Dynamically replaced by activity
  - This app: used whenever possible
3. Architecture: Overview

- Zeeguu
  - Zeeguu-ConnectionManager
  - ZeeguuAccount
- Feedly
  - FeedlyAccount
  - Feedly-ConnectionManager
- Database
  - Database-Fragment
- BaseActivity
- MainActivity
  - MyWords
  - FeedOverview
  - FeedEntryList
  - FeedEntry
- SettingsActivity
  - MainSettings
  - ZeeguuSettings
  - LoginDialog
3. Architecture: Back End

• **ConnectionManager**
  - Classes to communicate with Zeeguu and Feedly API
  - Uses Volley

• **Account**
  - Manages user data
  - **ZeeguuAccount**
    - Stored in SharedPreferences
  - **FeedlyAccount**
    - Handles synchronization
    - Interface to Database
3. Architecture: Back End

- ORM (Object Relational Mapping)
  - Implemented with ORMLite
  - Works with annotations
  - Uses DAO pattern (Data Access Objects)
  - Flexible QueryBuilder to easily construct queries
  - Does not directly support many-to-many relations
4. Article Recommender

- Helps the user to find suitable articles to read
- Presented in “Zeeguu Recommended” category
- Implemented on the Zeeguu server
- Two components
  - Difficulty
  - Learnability
4. Article Recommender: Idea

- Analyzes text on word-based level
- Two metrics used to estimate difficulty
  - KnownWordProbability
  - RankedWord (Word frequency lists)
- Problem: Shortened feed content
  - Goose content extractor
- Evaluation: Case study
5. Conclusion

• Conclusion
  • Zeeguu Reader makes it possible to learn a new language in a comfortable way on Android devices
  • Includes planned features, still room for extensions

• Personal Lessons Learned
  • ORM: Comfortable way to implement database
  • Prioritize planned features
  • Gained experience in new programming languages
  • Performance optimization
Questions?
Additional Material: ORM

```java
@DatabaseTable(tableName = "feeds")
public class Feed {

    // Id is generated by the database and set on the object
    @DatabaseField(generatedId = true)
    private int id;

    @DatabaseField(columnName = "favicon", dataType= DataType.BYTE_ARRAY)
    private byte[] favicon;

    /*
     * If eager is set to false then the collection is considered to be "lazy" and will iterate
     * over the database using the Dao.iterator() only when a method is called on the collection.
     */
    @ForeignCollectionField(eager = false, orderColumnName = "date", orderAscending = false)
    private ForeignCollection<FeedEntry> entries;  // one-to-many

    /*
     * Only for read access, categories stored in this list do not get saved in the database!
     * (Workaround because ormlite does not directly support m:m relations)
     */
    private ArrayList<Category> categories = new ArrayList<>();  // many-to-many
```
**Additional Material: ORM**

- **DAO Example**

```java
371     public void saveFeedEntry(FeedEntry entry) {
372         try {
373             if (entry.getId() == 0)
374                 feedEntryDao.create(entry);
375             else {
376                 feedEntryDao.update(entry);
377             }
```

- **Query Example**

```java
132     public List<FeedEntry> getRecommendedEntries(float maxDifficulty) {
133         try {
134             return callback.getDatabaseHelper().getFeedEntryDao().queryBuilder()
135                 .orderBy("zeeguu_difficulty_average", true)
136                 .where().between("zeeguu_difficulty_average", 0, maxDifficulty)
137                 .query();
138         }
139     catch (SQLException e) {
```
• Schema upgrade

```java
@Override
public void onUpgrade(SQLiteDatabase db, ConnectionSource connectionSource, int oldVersion, int newVersion) {
    try {
        Log.d(DatabaseHelper.class.getName(), "onUpgrade");

        // Drop the old tables
        TableUtils.dropTable(connectionSource, Category.class, true);
        TableUtils.dropTable(connectionSource, CategoryFeed.class, true);
        TableUtils.dropTable(connectionSource, Feed.class, true);
        TableUtils.dropTable(connectionSource, FeedEntry.class, true);

        // After we drop the old databases, we create the new ones
        onCreate(db, connectionSource);
    }
    catch (SQLException e) {
```

```java
/**
 * Database class to allow a many-to-many relation between categories and feeds in ormlite
 */
@DatabaseTable(tableName = "category_feed")
public class CategoryFeed {

    /**
     * This id is generated by the database and set on the object when it is passed to the create method. An id is
     * needed in case we need to update or delete this object in the future (ormlite does not support multiple
     * primary keys).
     */
    @DatabaseField(generatedId = true)
    private int id;

    // This is a foreign object which just stores the id from the Category object in this table.
    @DatabaseField(foreign = true, columnName = "category_id", columnDefinition = "integer references categories(id) on delete cascade")
    Category category;

    // This is a foreign object which just stores the id from the Feed object in this table.
    @DatabaseField(foreign = true, columnName = "feed_id", columnDefinition = "integer references feeds(id) on delete cascade")
    Feed feed;

    CategoryFeed() {
        // Empty constructor needed by ormlite
    }

    public CategoryFeed(Category category, Feed feed) {
        this.category = category;
        this.feed = feed;
    }
}
```
• Zeeguu WebView
  • Extended Android WebView
  • Allows translation & bookmarking of words
  • Injects JavaScript in every webpage
  • JavaScript to Java Interface

• How does it work?
  • Word selection extension
  • Submit word for translation
  • Bookmark: Extract context
  • Highlight bookmarked word(s) using regex
difficulties = []

for text in texts:
    # Calculate difficulty for each word
    words = util.split_words_from_text(text['content'])
    words difficulté = []
    for word in words:
        ranked_word = RankedWord.find_cache(word, language)

        word difficulté = 1.0  # Value between 0 (easy) and 1 (hard)
        if ranked_word is not None:
            # Check if the user knows the word
            try:
                known_probability = known_probabilities[word]  # Value between 0 (unknown) and 1 (known)
            except KeyError:
                known_probability = None

        if personalized and known_probability is not None:
            word difficulté -= float(known_probability)
        elif ranked_word.rank <= rank_boundary:
            word_frequency = (rank_boundary - (ranked_word.rank - 1)) / rank_boundary  # Value between 0 (rare) and 1 (frequent)
            word difficulté -= word_frequency

    words difficulté. append(word difficulté)
Additional Material: Evaluation

• Case study
  • Mircea as participant
  • 9 articles from different difficulty levels
  • Video recording, “think aloud”

• Analysis
  • Understanding
  • Time per character
  • Percentage of words looked up
  • Percentage of words bookmarked
### Results (Average for difficulty groups)

<table>
<thead>
<tr>
<th>Score</th>
<th>Understanding</th>
<th>Time per char</th>
<th>Looked up</th>
<th>Bookmarked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy (0.24)</td>
<td>4.50</td>
<td>0.21 s</td>
<td>6.52 %</td>
<td>5.19 %</td>
</tr>
<tr>
<td>Medium (0.32)</td>
<td>3.33</td>
<td>0.23 s</td>
<td>7.75 %</td>
<td>6.81 %</td>
</tr>
<tr>
<td>Hard (0.44)</td>
<td>2.66</td>
<td>0.28 s</td>
<td>11.13 %</td>
<td>7.92 %</td>
</tr>
<tr>
<td>O</td>
<td>P</td>
<td>Q</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----------------</td>
<td>--------------------------------</td>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>4.50</td>
<td>0.237906423</td>
<td>7.774390244</td>
<td>6.25</td>
<td></td>
</tr>
<tr>
<td>4.50</td>
<td>0.170936296</td>
<td>4.516129032</td>
<td>3.870967742</td>
<td></td>
</tr>
<tr>
<td>4.50</td>
<td>0.227593152</td>
<td>7.272727273</td>
<td>5.454545455</td>
<td></td>
</tr>
<tr>
<td>1.50</td>
<td>0.232</td>
<td>8.860759494</td>
<td>7.911392405</td>
<td></td>
</tr>
<tr>
<td>4.00</td>
<td>0.185257032</td>
<td>4.733727811</td>
<td>4.142011834</td>
<td></td>
</tr>
<tr>
<td>4.50</td>
<td>0.25974026</td>
<td>9.653916211</td>
<td>8.378870674</td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>0.318133616</td>
<td>19.38534279</td>
<td>13.23877069</td>
<td></td>
</tr>
<tr>
<td>4.00</td>
<td>0.183260611</td>
<td>5.778894472</td>
<td>5.527638191</td>
<td></td>
</tr>
<tr>
<td>3.00</td>
<td>0.332525742</td>
<td>8.214285714</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4.5</td>
<td>0.21214529</td>
<td>6.521082183</td>
<td>5.191837732</td>
<td></td>
</tr>
<tr>
<td>3.333333333</td>
<td>0.225665764</td>
<td>7.49467839</td>
<td>6.810758304</td>
<td></td>
</tr>
<tr>
<td>2.666666667</td>
<td>0.277973323</td>
<td>11.12617433</td>
<td>7.922136292</td>
<td></td>
</tr>
</tbody>
</table>