ONIP-1 Better language model support

Status

Current State: Under Discussion

Discussion Thread: https://lists.apache.org/thread.html/50ac99671c0343980053e7ba3faa3e4c2624b08b1f033617f222c33e@%3Cdev.opennlp.apache.org%3E

JIRA: OPENNLP-1183

- **Motivation**: Language models can be useful in typical NLP tasks like spellchecking, suggestions (and NLG in general) but also as building blocks in other areas like in Machine Translation or Information Retrieval. Current *LanguageModel* API, despite some deprecated methods which might be removed in future releases, looks ok. However the only existing implementation *NGramLanguageModel* has some outstanding limitations with respect to :
    - memory consumption - each ngram is stored as a _Map<StringList,Integer>_ (each StringList being an ngram and the int being the count of each ngram in the dictionary). Although that's better than using *String[]* as keys, it still consumes quite some memory when compared to other libraries (e.g. KenLM). Also there is no caching or precomputing of probabilities, so probability estimation can be very slow at runtime.
    - space consumption - *NGramLanguageModel* currently extends from *NGramModel* and inherits its serializing capabilities as an XML, that is very expensive when compared to Model serialisation algorithms for other OpenNLP components and LM libraries and makes it uncomfortable to use with very large dictionaries.
    - accuracy / perplexity - currently *NGramLanguageModel* estimates probability using *Stupid Backoff* algorithm, (combinations of) other alternatives might work better (e.g. Kneser Ney).

- **Proposed Changes**: Here're the proposed changes:
    - create a new implementation of *LanguageModel* API which extends from *BaseModel* and where we can better handle model serialization algorithm
    - use hashing techniques to store ngrams instead of the current *Map<StringList, Integer>*
    - evaluate techniques for caching / prefetching of probabilities
    - implement Kneser-Ney smoothing and make it possible to plug new techniques for probability estimation

- **New or Changed Public Interfaces**: no backward compatibility concerns should arise given that would be a new implementation of an existing API, removing the currently deprecated methods in LM API is a concern that should be handled separately from this improvement

- **Migration Plan and Compatibility**: see above

- **Rejected Alternatives**: changing the current *NGramLanguageModel* implementation would be possible but would pose backward compatibility issues, additionally by keeping the old implementation we can measure the improvements by comparing space / runtime storage efficiency and accuracy / perplexity between the current and the new implementation more easily.

- **Useful resources**: https://web.stanford.edu/~jurafsky/slp3/4.pdf