SCALE: A Scalable Language Engineering Toolkit

Joris Pelemans[1], Lyan Verwimp[1], Kris Demuynck[2], Hugo Van hamme[1], Patrick Wambacq[1]
[1]ESAT, KU Leuven, Belgium
[2]ELIS, Ghent University, Belgium
{joris.pelemans,lyan.verwimp,hugo.vanhamme,patrick.wambacq}@esat.kuleuven.be
kris.demuynck@elis.ugent.be

Abstract

In this paper we present *SCALE*, a new Python toolkit that contains two extensions to n-gram language models. The first extension is a novel technique to model compound words called Semantic Head Mapping (SHM). The second extension, Bag-of-Words Language Modeling (BagLM), bundles popular models such as Latent Semantic Analysis and Continuous Skip-grams. Both extensions scale to large data and allow the integration into first-pass ASR decoding. The toolkit is open source, includes working examples and can be found on `http://github.com/jorispelemans/scale`.

Keywords: language engineering, compound modeling, bag of words

1. Introduction

Although recurrent neural networks (RNNs) have recently become the state of the art in language modeling (Mikolov et al., 2010), they do not yet scale as well to large data as n-gram language models do and are as such impractical for automatic speech recognition (ASR). Even after optimization, an RNN language model (LM) is too computationally expensive to be applied directly during decoding and its use is therefore restricted to a multi-pass approach in which the speech signal is first decoded using a simpler (often n-gram) LM, after which the output (N-best lists or lattices) is rescored using the RNNLM.

In this paper we present *SCALE*, a new Python toolkit that contains two extensions to n-gram LMs: Semantic Head Mapping (SHM) and Bag-of-Words Language Modeling (BagLM). Both of them scale to large data and allow the integration into first-pass ASR decoding. The toolkit is open source, includes working examples and can be found on `http://github.com/jorispelemans/scale`.

2. SHM for compound modeling

One of the main issues in language modeling is data sparsity: there is not enough training material to derive reliable statistics for every possible word sequence. This is in part caused by the ongoing formation of new words, a process that has only accelerated with the rise of the Internet e.g. Netizen, tweeple, infobesity. To successfully model these words, it is essential to look at lexicological processes and investigate the mechanisms that underly the formation of words.

One of the most productive mechanisms in many languages is compounding, which induces the frequent creation of numerous new words all over the world. In this section we briefly describe SHM: a tool to find semantic heads for compounds. We also present some results that were achieved using this tool. For more details we refer the reader to (Pelemans et al., 2014a) and (Pelemans et al., 2015).

2.1. Semantic Head Mapping

In recent work (Pelemans et al., 2014a; Pelemans et al., 2015) we presented a novel clustering technique for compound words, called Semantic Head Mapping. We argue that compounds are well represented by their semantic heads and as a consequence that compound-head clusters do not suffer as much from overgeneralization as other clustering techniques. By mapping compounds onto their semantic heads, the technique is able to estimate n-gram probabilities for infrequent and even unseen compounds. This approach is especially interesting for domain adaptation purposes where new, domain specific words are introduced into the vocabulary, but can also be applied in more general contexts. The technique was evaluated on Dutch, but the idea may extend to languages with similar compound formation rules.

Because existing morphological information is rarely available for infrequent words, the SHM tool was built from scratch and consists of two parts: (1) a generation module which generates all possible decompounding hypotheses; and (2) a selection module which selects the most plausible head.

2.2. Generation module

First, all possible decompounding hypotheses are generated by means of a brute-force lexicon lookup: for all possible substrings w_1 and w_2 of the candidate compound w, $w = w_1 + w_2$ is an acceptable hypothesis if w_1 and w_2 are both in the lexicon. The substrings are optionally separated by binding morphemes. The module works recursively on the first substring i.e. if w_1 is not in the lexicon, the module will verify whether or not it is a compound by itself. In its current implementation, the system always makes the assumption that the head is located at the right-hand side of the compound, since this is almost exclusively the case for Germanic languages.

To account for possible discrepancies between modifiers and heads, we allow the generation module to read from

This research is funded by the Flemish government agency IWT (project 130041, SCATE) and by IWT–INNOVATIEF AANBESTEDEN and VRT in the STON project.

two different lexica: a modifier lexicon V_m and a head lexicon V_h. The user can decide how to create these lexica e.g. by word frequency. Every hypothesis that has a modifier or head that does not occur in the corresponding lexicon, is automatically ignored. An exception is made for acronym modifiers consisting of all uppercase characters, which are automatically considered as valid words and are therefore not required to be lexical.
Some additional filtering is possible based on constituent length. Allowing short constituents often results in a drastic decrease in precision, especially if the lexica contain (noisy) infrequent short words. Two parameters L_m and L_h are introduced to control the minimal length of modifiers and heads respectively.

2.3. Selection module

The generation module hugely overgenerates because it only has access to lexical knowledge. In the selection module we introduce knowledge based on corpus statistics to select the most likely candidate. Concretely, the choice between the remaining hypotheses is based on unigram probabilities and constituent length: we provide selection parameters w_{len}, w_u and w_{pu} to weigh the relative importance of the head length, head unigram probability and product of the constituent unigram probabilities. We also allow the use of part-of-speech (POS) knowledge, although POS tags are often incorrect for infrequent compounds. The toolkit allows the exploitation of POS knowledge in two different ways: by constraining the head to have the same POS as the compound or by specifying compound rules e.g. only allow noun + noun.
Algorithm 1 shows pseudocode for the complete SHM algorithm, excluding binding morphemes and POS knowledge for the sake of clarity.

function GENERATE($compound, V_m, V_h, L_m, L_h$)
 for all $mod + head = compound$ **do**
  if $len(mod) \geq L_m$ **and** $len(head) \geq L_h$ **then**
   if $head \in V_h$ **then**
    if $mod \in V_m$ **or** $mod\ is\ acronym$ **then**
     $hypotheses \leftarrow (mod, head)$
    else
     $hypotheses \leftarrow (\text{GENERATE}(mod, ...), head)$
 return $hypotheses$
function SELECT_BEST($hypotheses, w_{len}, w_u, w_{pu}$)
 for all $(mod, head) \in hypotheses$ **do**
  score $\leftarrow w_{len} * len(head) + w_u * P_{uni}(head)$
   $+w_{pu} * P_{uni}(mod) * P_{uni}(head)$
  if $score > max\_score$ **then**
   $max\_score \leftarrow score$
   $best \leftarrow (mod, head)$
 return $best$

Algorithm 1: Semantic head mapping algorithm

2.4. Results

Table 1 shows some results that were achieved using SHM on a test set extracted from the *Corpus Spoken Dutch* (*CGN*) (Oostdijk, 2000). It can be seen that the tool is capable of finding the semantic head with a precision and recall of ca. 80% and 86% respectively. Using the probability estimation techniques discussed in (Pelemans et al., 2014a), it is possible to adapt a baseline n-gram LM to incorporate unseen compounds which gives an absolute reduction in word error rate (WER) of ca. 1%. For more information and results we refer the reader to (Pelemans et al., 2014a) and (Pelemans et al., 2015).

			WER	
	Precision	Recall	n=3	n=5
no mapping	-	-	28.23%	27.53%
SHM	80.25%	85.97%	27.29%	26.62%

Table 1: Semantic Head Mapping (SHM) results on a test set from the *Corpus Spoken Dutch* (*CGN*). Word error rates (WERs) are compared to a baseline n-gram LM (no mapping).

3. BagLM for long-distance modeling

The main weakness of n-gram LMs is their primary assumption that the history upon which the prediction is based, can be reduced to only a handful of words. Although it may be the case that much if not most of the information resides in the immediate, local context of the current word (Rosenfeld, 1994), other long-span phenomena such as sentence- or document-level semantic relations can only be modeled with the help of the more distant history. One of the ways to address this weakness is to combine n-gram LMs with so-called *bag-of-words* techniques that model word similarity in the hope of capturing both local and global phenomena.
The focus of this section is *BagLM*: a tool that provides several of these techniques including well established models such as cache models (Kuhn and de Mori, 1990) and models based on Latent Semantic Analysis (Deerwester et al., 1990). New is a bag-of-words LM that we recently proposed (Pelemans et al., 2014b) which is based on the continuous skip-gram model (Mikolov et al., 2013b; Mikolov et al., 2013a; Mikolov et al., 2013c). The cache model was implemented from scratch, whereas the other models are wrapped around existing implementations of similarity models from the open-source Python framework *gensim* (Řehůřek and Sojka, 2010). In what follows we briefly describe the three main models of the tool and present some results that were achieved using the tool. For more details on the models and experiments we refer the reader to (Pelemans et al., 2014b).

3.1. Cache models

Cache models are based on the observation that topical words tend to re-occur within a text. A cache memory is kept that keeps track of the last K words in a document and is consulted when predicting the next word. In their simplest form (Kuhn and de Mori, 1990), cache models distribute the probability mass uniformly among all the K tokens in the cache memory:

$$P_{cache}(w_q|w_{q-K}^{q-1}) = \frac{C_{cache}(w_q)}{K} \quad (1)$$

where $C_{cache}(w_q)$ indicates the frequency of word w_q in the cache memory.
Cache models are simple and efficient and are capable of modeling long distance phenomena which is the main weakness of n-gram LMs. They do not however employ any kind of semantic knowledge.

3.2. Latent Semantic Analysis

Latent Semantic Analysis (LSA) (Deerwester et al., 1990) is one of the earliest attempts to discover hidden semantic structure in a text by considering word co-occurrences. It is a dimensionality reduction technique based on truncated singular value decomposition that is applied on a term-document matrix W which contains in each cell the number of times a word (rows) occurs in a document (columns). The information in W that corresponds to the smallest singular values is considered to be noise induced by data errors and word redundancy and is effectively removed by preserving only the k largest singular values. The resulting rank k approximation is optimal with respect to the Frobenius norm and uncovers latent semantic relations between words and documents.

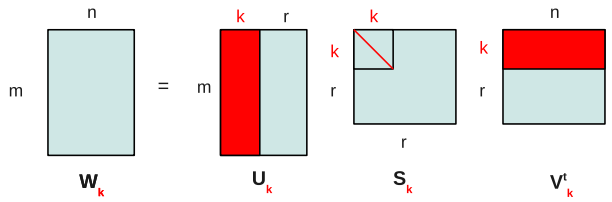

Figure 1: (Truncated) Singular Value Decomposition

Often a preprocessing step is useful, because the documents are not of equal length and not all words are equally informative. To this end, the raw counts of W may be transformed according to a weighting scheme which typically consists of a global component $G(i)$ and a local component $L(i, j)$ (Dumais, 1991). The same global weight – indicating the overall importance of a term – is applied to an entire row of the matrix, whereas the local weight – which indicates the importance of a term in a specific document – is applied to each cell in the matrix.
Although many different schemes exist, the choice is not so critical to the overall performance of LSA. TF-IDF is one of the most used weighting schemes in the context of information retrieval and also one with which we have had good results. For an overview and classification of different weighting schemes, we refer to Salton (1971).
Semantic similarity between documents and words is measured by calculating the cosine distance in the latent space. In the context of language modeling, the history is considered to be a (pseudo-)document $\tilde{d}$ and the cosine distance K between the word vector u_q and the (pseudo-)document vector $\tilde{v}_{q-1}$ in the latent space is converted into a probability as follows (Coccaro and Jurafsky, 1998):

$$P(w_q|\tilde{d}) = \frac{[K(u_q, \tilde{v}_{q-1}) - \min_u K(u, \tilde{v}_{q-1})]^\gamma}{\sum_{w_i \in \mathcal{V}} [K(u_i, \tilde{v}_{q-1}) - \min_u K(u, \tilde{v}_{q-1})]^\gamma} \quad (2)$$

where $\mathcal{V}$ is the vocabulary and γ is a parameter that controls the dynamic range of the distribution.
Although LSA is capable of uncovering semantic relations between words and documents, it is insensitive to the multiple senses that many words have. Moreover, using the Frobenius norm as an error function assumes normally distributed data with independent entries, which is not the case for the counts in the term-document matrix.

3.3. Continuous skip-gram model

Motivated by the recent successes in neural network language modeling, Mikolov et al. (Mikolov et al., 2013b; Mikolov et al., 2013a; Mikolov et al., 2013c) recently proposed the continuous skip-gram model (CSM): a new architecture for the acquisition of high-quality word embedding vectors which might not be able to represent the data as precisely as neural networks, but is less complex and can therefore handle more data efficiently. CSM is a log-linear classifier with a continuous projection layer that tries to maximize the prediction of words within a range R before and after the current word. The network is trained by using backpropagation with stochastic gradient descent until convergence and uses the softmax activation function to ensure that the output layer forms a valid probability distribution:

$$p(w_{t+j}|w_t) = \frac{\exp(v_{w_t}^T v'_{w_{t+j}})}{\sum_{w=1}^{W} \exp(v_{w_t}^T v'_{w_{t+j}})} \quad (3)$$

where v_w and v'_w are the input and output representations of word w.
The model was made more efficient by approximating the full softmax by a hierarchical version which was first introduced by Morin and Bengio (2005). The hierarchical softmax uses a binary Huffman tree representation of the output layer with the words as its leaves. It assigns short binary codes to frequent words which has a significant positive effect on the overall speed of the model. For more information on the hierarchical softmax and the continuous skip-gram model in general, we refer the reader to Mikolov et al. (2013a) and Mikolov et al. (2013b).

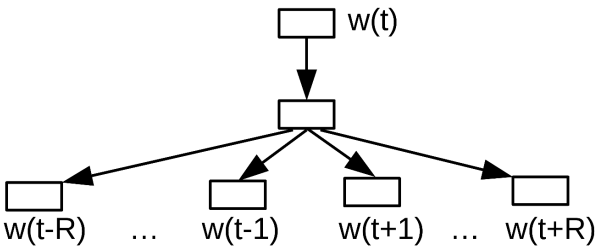

Figure 2: Continuous Skip-gram Model

An interesting observation was made in Mikolov et al. (2013c): the vector-space word representations that are implicitly learned by the input-layer weights are surprisingly good at capturing both semantic and syntactic regularities in language and each relationship is characterized by a relation-specific vector offset. This allows intuitive vector mathematics based on the offsets between words e.g. *king* - *man* + *woman* results in a vector that is very close to that of

queen. It is clear that these CSM word embedding vectors can be a valuable source of information to enrich existing LM techniques.

For word embedding vectors to be incorporated into a LM, we need to have a representation at a higher level than just individual words. In the current implementation we chose to calculate the centroid of the previous K word vectors, even though this makes the unreasonable assumption that the meaning of a phrase is equal to the sum of its components. We then calculated the cosine distance of this centroid to the predicted word, using a similar technique as Eq. (2) to end up with a probability distribution.

Mikolov et al. (2013b) show that the assumption that the meaning of a phrase is equal to the sum of its components can be overcome in part by detecting common phrases i.e. words that appear frequently together and infrequently in other contexts. This way they were able to replace *New York Times* by a single token while *this is* remained unchanged. We have not experimented with this approach, but the toolkit certainly allows it.

They did not however address the assumption of equal word importance. Function words like *the* and *of* are clearly less informative than content words, hence should be given less weight. This can be dealt with quite easily by applying the same TF-IDF weighting as was mentioned in Section 3.2..

3.4. Results

Table 2 shows some results that were achieved using various combinations of n-grams with BagLM models on a test set extracted from the Flemish magazine *Knack*. It can be seen that the BagLM models are a valuable extension of n-grams with perplexity reductions of up to ca. 13% and 21% for individual and multiple models, respectively. For more information and results we refer the reader to (Pelemans et al., 2014b).

	PPL	Reduction
3-gram	210.36	
4-gram	198.10	5.83%
5-gram	197.69	6.02%
3-gram+cache	187.89	10.68%
3-gram+LSA	186.12	11.53%
3-gram+CSM	183.62	12.71%
3-gram+cache+CSM	175.82	16.42%
4-gram+cache+CSM	166.20	21.00%
5-gram+cache+CSM	166.16	21.02%

Table 2: Perplexity results for various combinations of n-gram and BagLM models using a vocabulary of 50,000 words, as measured on a test set from the Flemish magazine Knack.

4. Conclusion

We presented a new toolkit that contains two extensions to n-gram language models: SHM and BagLM. Both of them scale to large data and allow the integration into first-pass ASR decoding. The toolkit is open source, includes working examples and can be found on `http://github.com/jorispelemans/scale`. To the best of our knowledge, no other toolkits exist that provide this functionality.

5. Bibliographical References

Coccaro, N. and Jurafsky, D. (1998). Towards Better Integration Of Semantic Predictors In Statistical Language Modeling. In *Proc. ICSLP*, pages 2403–2406.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Dumais, S. T. (1991). *Behavior Research Methods, Instruments, & Computers*, (2):229–236.

Kuhn, R. and de Mori, R. (1990). A Cache-Based Natural Language Model for Speech Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(6):570–583.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Proc. Interspeech*, pages 1045–1048.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed Representations of Words and Phrases and their Compositionality. In *Proc. NIPS*, pages 3111–3119.

Mikolov, T., tau Yih, W., and Zweig, G. (2013c). Linguistic Regularities in Continuous Space Word Representations. In *Proc. HLT-NAACL*, pages 746–751.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Proc. AISTATS*, pages 246–252.

Oostdijk, N. (2000). The Spoken Dutch Corpus. *The ELRA Newsletter*, 5(2):4–8. http://lands.let.ru.nl/cgn/.

Pelemans, J., Demuynck, K., Van hamme, H., and Wambacq, P. (2014a). Coping with Language Data Sparsity: Semantic Head Mapping of Compound Words. In *Proc. ICASSP*, pages 141–145.

Pelemans, J., Demuynck, K., Van hamme, H., and Wambacq, P. (2014b). The effect of word similarity on N-gram language models in Northern and Southern Dutch. *CLIN*, 24:91–104.

Pelemans, J., Demuynck, K., Van hamme, H., and Wambacq, P. (2015). Improving N-gram Probabilities by Compound-head Clustering. In *Proc. ICASSP*, pages 5221–5225.

Řehůřek, R. and Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proc. LREC*, pages 45–50.

Rosenfeld, R. (1994). *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Ph.D. thesis, Carnegie Mellon University.

Salton, G. (1971). *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Inc.