Going Neurotic with Neural Word Embeddings

Luís Sarmento

luis.sarmento@gmail.com
https://www.linkedin.com/in/luissarmento/

June 17, 2017

Introduction

What are Word Embeddings?

- Vector *representations* of words with desirable properties:
  - encode as much information
    - semantic / grammatical as possible
  - compact / “low” dimensional
  - have a convenient topology (being a metric space)
- 1-hot vector over a lexical space is a vector representation
  - It is **not** what we consider an embedding
  - It only encodes the information:
    - “we are in the presence of this token”
    - it is very high dimensional (size of vocab)
    - there is no “metric” (all vectors are disjoint)

What are Word Embeddings?

- We want the embedding vector to encode information about the semantics of the word
  - "this word has something to do with geo-location"
  - "this word tends to show up in positive emotion contexts"
- Not all of the aspects need be so explicit or easy to read
- Knowing all these aspects about a word helps in NLP tasks
  - e.g. you can use this embedded information as "features"

This is not such an exotic concept

- Suppose we have set of Lexical-semantic dictionaries (manually built)
    - word - POS class
    - word - sentiment class
    - word - geo-gazetteer
    - word - name / surname
    - ...

  or even some statistical information about words
    - word - probability of showing up in 1 position a sentence
    - word - probability of being followed by a number

This is not such an exotic concept

- If we build a vector for each word such as

$$v(w_i) = [\ \mathrm{POS}(w_i),\ \mathrm{SNT}(w_i),\ \mathrm{GEO}(w_i),\ \mathrm{NM}(w_i)\ \ldots\ P_1(w_i),\ P_{\#}(w_i)\ \ldots\ ]$$

we have a sort of word embedding!

  - although we have a really weird "space topology", with no "intuitive" notion of distance

But we want to go one step further

- We want to **learn** these embeddings from data
  - Human encoding does not scale that well
  - There are many aspects which are hard to encode and we many need machine to help us
    - Some aspects are continuous / multidimensional
  - There are linguistic biases we want to avoid
    - Being data-driven may help to reduce such bias
- And we want to **transfer** what we learn from one task, to speed up training systems for **another** task

But there is more...

- We want to understand the limits of those representations:
  - what they encode (or can't possibly encode!)
  - how they encode
- We want to understand how to engineer representations
  - how to encode more complex information relations (e.g. hierarchies, cause-effect, etc)

But there is more...

- More than just a practical question for training NLP systems
- Concept representations are essential for the operation of "intelligent systems"
- Understanding these representations may helps us understand how we - humans - process language and operate on concepts

Learning Neural Word Embeddings

Basic Idea

- Think about a ML task (involving words[1])
- Learn a neural model for that task:
  - restricted to some vocabulary
- Extract the kernel matrices of specific layers
- Use those matrices as look up tables of the embedding vectors (for the vocabulary)

[1] In this context, a "word" is understood as "token". Lexical compounds such as "New York City" are understood as 3 words / tokens.

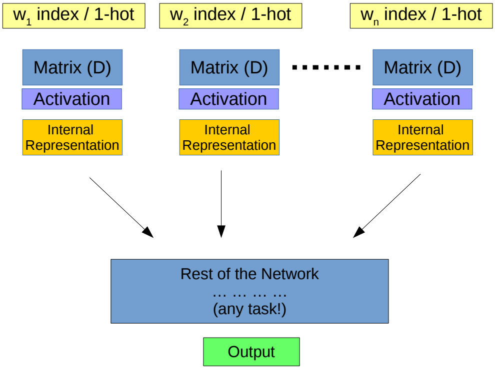
A bit more detail

- Given $< X, Y >$ pairs
  - At least one of the X or Y is a “word” / text
- We are going to learn a ANN that maps X to Y:
  - X = words, Y = some media (sound, words, images)
  - X = some media, Y = words
- The ANN will transform the signal from X to Y
  - there are intermediate representations in the hidden layers
  - word embeddings can be found on these representations

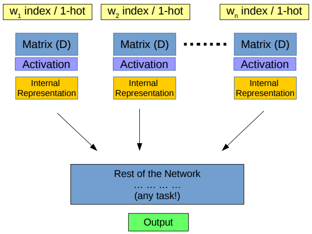
What layers?

- The layers that actually hold word embeddings are:
  - layers that take a word (actually word indexes) as input and map it to an internal representation
    - Input (Direct) Embeddings
  - layers that take internal representation and produce / output a word (actually word index)
    - Output (Indirect) Embeddings

Input (Direct) Embeddings

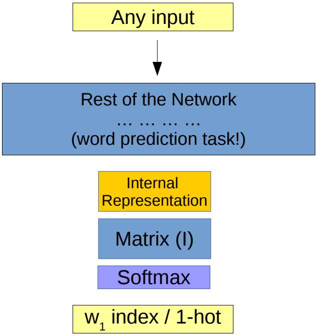


Input (Direct) Embeddings

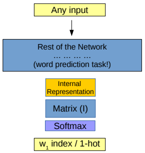


- D matrix maps word indexes to internal representations
- the embedding information is in the D matrix

Output (Indirect) Embeddings



Output (Indirect) Embeddings



- I matrix maps internal representation to a (distribution of) indexes of over Y
- the embedding information is in the I^T matrix, which maps indexes to internal representations

Important Intuition

- Assuming that ANN(X) = Y is "generalizing well" (not just memorizing everything!)
- If several Y's are possible for the same/similar X's:
  - the upstream / hidden representations for these Y's are same/similar
  - the Y's have similar/close embeddings
- All this assuming we have a way of measuring closeness
  - e.g. cosine of vectors

It's a simple concept but...

- ... in practice there are still many possibilities
  - Which task? What Input to Output?
  - For a given task, which model architecture?
  - For each model architecture, which parameters?
- How do we find the best way to create embeddings?
- How do we evaluate the resulting embeddings?

Which task?

- It depends on what information we want to embed
  - Voice to Text
    - similar sounds will produce "similar"
    - We are embedding phonetic information
    - E.g. "two" / "to" / "too"
      (assuming that the sound are actually "similar")
  - Text to Text:
    - similar meanings / semantics, will produce similar words (ah!)
    - We are embedding some sort of semantic information
    - E.g. "porque" / "pq", "lisbon" / "city", ...
    - ...
- In this talk, we will focus on Text-Text spaces
  - "easier" to work with (less formats to process)

Supervised vs Unsupervised?

- But it also depends on the data we have X / Y available
  - Supervised Setting:
    - We have X and Y
    - Manual annotation, or annotation induced from logs
  - Unsupervised Setting:
    - We only have X: we have to create (X, Y) pairs just from X
    - Cheap!! Tons of data to learn from!
- We are going to focus on the unsupervised space!

Creating X $\rightarrow$ Y tasks: some examples

- Totally unsupervised: use snippets of text
  - Take N words (X), try to predict the next (Y)
  - Take a window of N words around a central word (X), try to predict the central word (Y)
  - Take a window of N words, shuffled them and take one out (X), try to predict the missing word (Y)
  - Or do the reverse. E.g. : take a word (X) and try to predict the N words on both sides (Y)
  - ...
- Pseudo-supervised: use any text to text pairings
  - Take snippets of Wikipedia abstracts (X), predict title / categories (Y)
  - Take a product title (X), predict the query (Y)

What model?

- There are several *explicit* word embedding models out there
  - Neural Word Prediction (Bengio et al)
  - Word2Vec (Mikolov et al)
  - Glove (Pennington et al)
- We also want to experiment with different types of models (starting from the simplest):
  - We are using Keras for building models
- More on this in next section...

How do we evaluate?

- There are two main strategies:
  - Intrinsic Evaluation:
    - We expect the embedding to encode some semantic relation (e.g. class similarity)
    - We test the presence of that relation directly using some "simple" measure / test
  - Indirect Evaluation:
    - We assume that the embeddings carry "valuable" semantic information
    - We use that information as features while training a model for a downstream task (e.g. sentiment classification)
    - We measure the impact of using the embeddings as features

But this is much trickier than it seems...

- Intrinsic Evaluation. It's hard to:
  - design an embedding to capture a precise relation we want
  - (for us humans) agree on gold standard for that relation
  - find meaningful measures over the vector space of embeddings:
    - cosine, KL-Divergence, Euclidean...
  - designing the test:
    - is anything above a threshold "good" and below it "bad"
    - do we only care about relative positions / ranks

But this is much trickier than it seems...

- Indirect Evaluation
  - The downstream model has it's own degrees of freedom (which may deeply affect the final result)
  - There may be some undesirable dilution / amplification of certain properties
  - what helps a specific downstream task may not help other downstream tasks: are the embeddings "generic"?
  - what help this downstream task may not be "desirable" in terms of re-usable knowledge representation

This is why I am going neurotic...

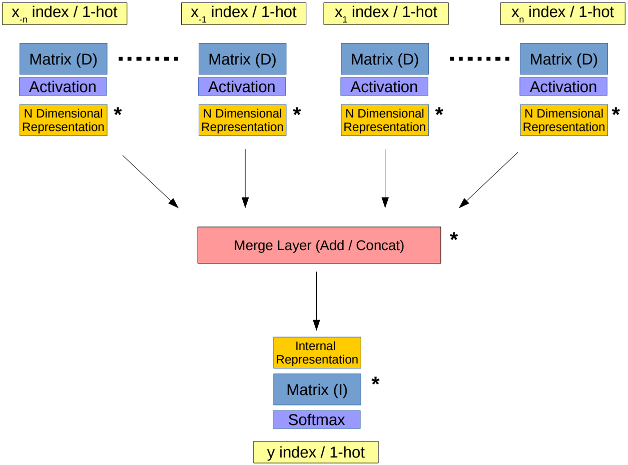
- Lot's of variables / parameters
- And on top of all this there are still
    - different model architectures (type of layers, depth, ...)
    - model hyper-parameters to tune
    - # of dimensions of the embedding space
    - how much data do we really need
    - all training parameters
- Can we learn how to design specific representations?
- Can we make any *systematic* progress on this?

One small step to man...

Trying to be systematic

- Syntagma:
  - Framework for experimenting with neural models for NLP tasks
  - Python and Keras for neural modeling
  - MySQL to store corpora and intermediate data
  - Some neural models already prepared for some tasks
    - Starting with: word embedding models
    - Next steps: low level tasks (tokenization, phrase detection)
- Main goals: speed up (systematic) experimentation!

The model we will be experimenting with



The model we will be experimenting with

- CBOW-like
- This model should be able to capture distributional similarity

Words that tend to occur in the same contexts are "similar"

- "na cidade de X"
  - X = [Lisboa, Paris...]
  - X = [onde,...]
  - X = [origem, ...]
- "na cidade de X em janeiro de"
  - X = [Lisboa, Paris...]

The model we will be experimenting with

- This model should be able to capture distributional similarity

Words that tend to occur in the same contexts are "similar"

- The model should place in "close" places:
    - synonyms (but also antonyms)
    - elements of same "class"
    - elements of related by "is-a" relations

(but will not be able to differentiate between these cases)

The model we will be experimenting with

- It is a relatively simple model:
  - It is quite shallow
  - No convolutions, no recurrent layers
  - No multi-task outputs
- But it already has a few parameters:
  - N-gram window (5,7,...)
  - Dimensionality of the input embedding layer
  - Merging function of the input representations (concat, addition, etc...)
  - Regularization(s) at several layers
  - Dimensionality of the output embedding
  - Size of the input and output vocabularies

What we did

- Took a large Twitter corpus (300M)
- Sampled 2M 7-grams (from about 2G)
- Prepared an X,Y training
  - Shape: $x_{-3}, x_{-2}, x_{-1}, y, x_1, x_2, x_3$
  - tokenization by white space (baseline)
- Dimensionality of the input lexicon (x_i) is **32k**
  - large enough to provide diversity
- Dimensionality of the output lexicon (y) is **4k**
  - small, but enough for the purpose

What we did

- Trained the model over 42 combinations of 3 parameters:
  - input-embedding-dim = [8, 16, 32, 64, 128, 256, 512]
  - merge-mode = [add, concat]
  - l1-weight = [0.0000, 0.0001, 0.001]
- Stored all the embeddings in a DB
- Used T-SNE to visualize the results
- See if we can get some intuition

Some Results

Some Results

- We will be presenting two things:
  - Nearest Neighbor Examples (sorted by cosine similarity)
    - Is this even working?
  - T-SNE maps (high-level observation)
    - What impact do the parameters (seem to) have?

NN of “:)”

1. (442, 0.569) ;)
2. (615, 0.509) :d
3. (451, 0.480) xd
4. (1178, 0.471) :-)
5. (570, 0.450) :(
6. (161, 0.448) !
7. (1004, 0.402) :p
8. (332, 0.389) .
9. (8, 0.375) LINK
10. (301, 0.367) ?
11. (425, 0.364) <3
12. (412, 0.360) lol
13. (3529, 0.357) ;-)
14. (935, 0.357) ahah
15. (737, 0.347) !!

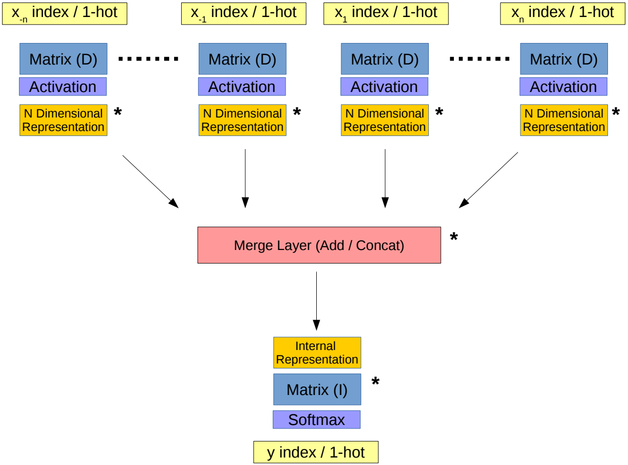
NN of "quatro"

1. (809, 0.536) três
2. (2617, 0.482) seis
3. (1637, 0.474) cinco
4. (608, 0.418) duas
5. (3305, 0.407) sete
6. (284, 0.371) dois
7. (2623, 0.368) 200
8. (2365, 0.352) várias
9. (2976, 0.345) dez
10. (2112, 0.340) próximos
11. (1012, 0.329) 14
12. (141, 0.321) 3
13. (307, 0.319) 10
14. (828, 0.317) 12
15. (2469, 0.317) vários

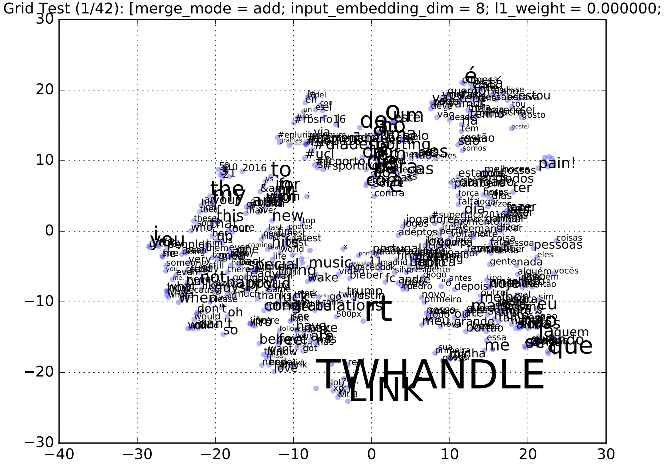
NN of “benfica”

1. (154, 0.524) sporting
2. (92, 0.444) porto
3. (2779, 0.375) scp
4. (1149, 0.358) benfica,
5. (2426, 0.356) fcp
6. (999, 0.352) benfica.
7. (610, 0.345) roma
8. (1643, 0.342) inter
9. (2965, 0.328) moreirense
10. (1369, 0.314) braga
11. (1006, 0.313) slimani
12. (2115, 0.311) sporting,
13. (1042, 0.307) arouca
14. (3343, 0.306) besiktas
15. (2326, 0.304) tondela

Playing with some parameters



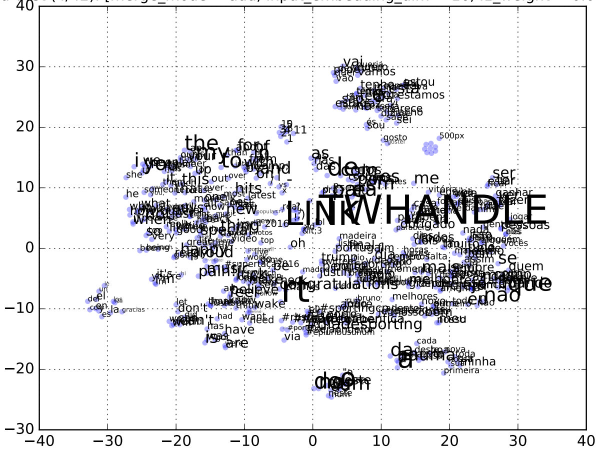
merge-mode=add, input-emb-dim=**8**, l1-weight=0.00



{"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 10

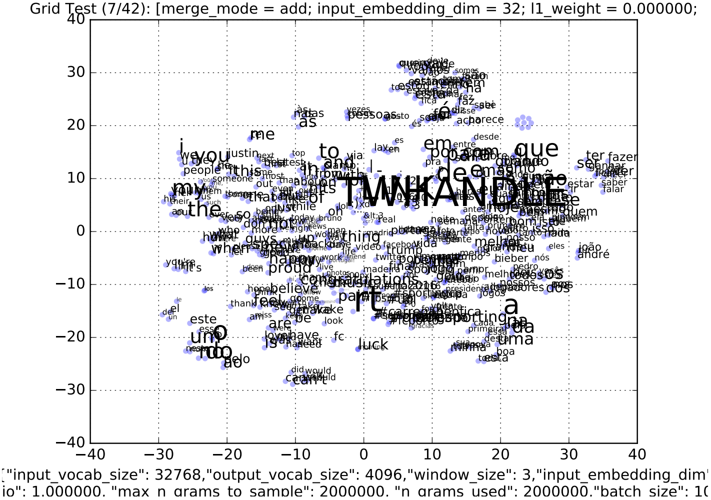
merge-mode=add, input-emb-dim=**16**, l1-weight=0.00

Grid Test (4/42): [merge_mode = add; input_embedding_dim = 16; l1_weight = 0.000000;



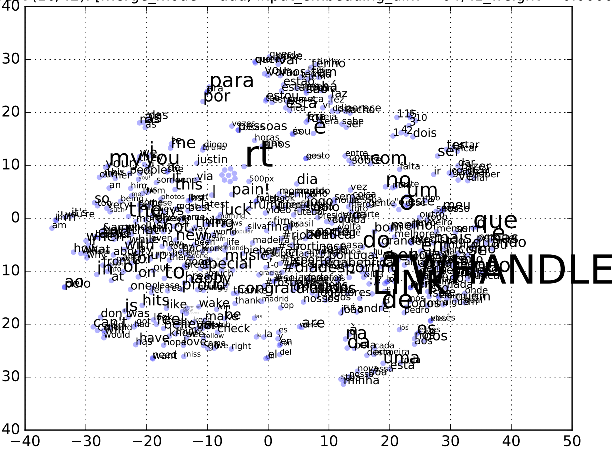
["input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 10

merge-mode=add, input-emb-dim=**32**, l1-weight=0.00



merge-mode=add, input-emb-dim=**64**, l1-weight=0.00

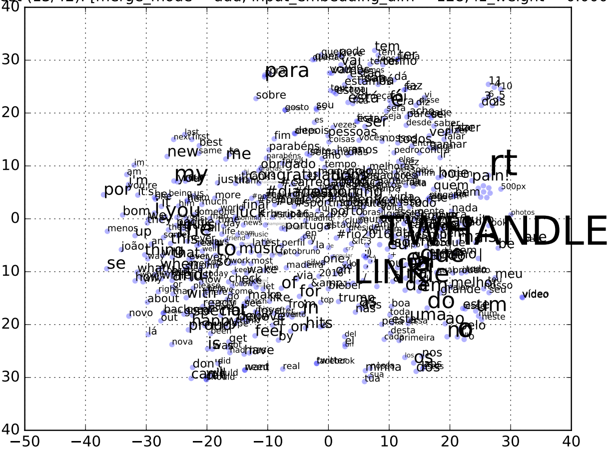
Grid Test (10/42): [merge_mode = add; input_embedding_dim = 64; l1_weight = 0.000000;



["input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 10

merge-mode=add, input-emb-dim=**128**, l1-weight=0.00

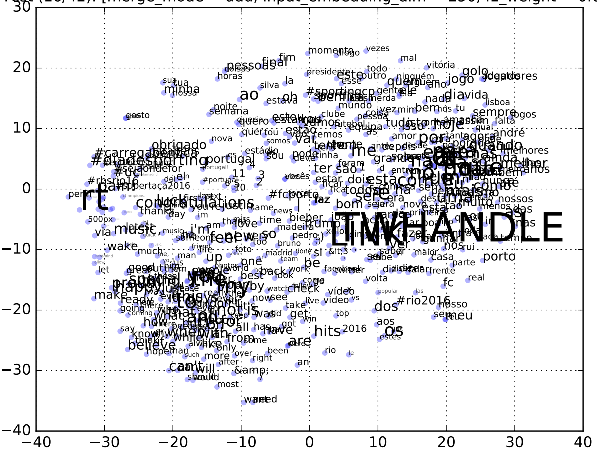
Grid Test (13/42): [merge_mode = add; input_embedding_dim = 128; l1_weight = 0.000000;



"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

merge-mode=add, input-emb-dim=**256**, l1-weight=0.00

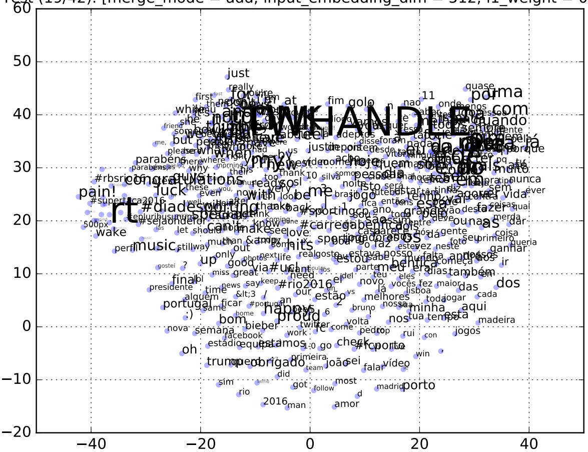
Grid Test (16/42): [merge_mode = add; input_embedding_dim = 256; l1_weight = 0.000000;



"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

merge-mode=add, input-emb-dim=**512**, l1-weight=0.00

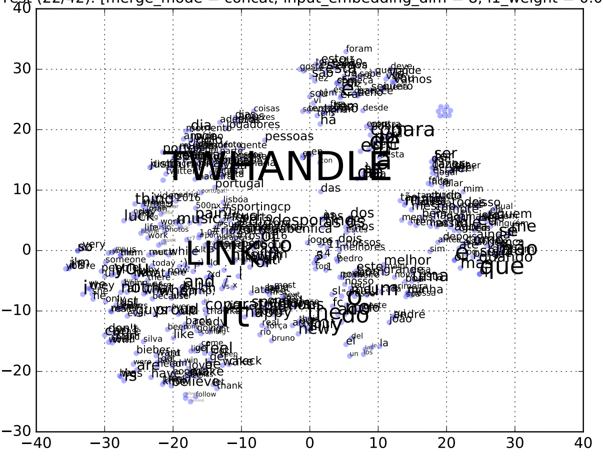
Grid Test (19/42): [merge_mode = add; input_embedding_dim = 512; l1_weight = 0.000000;



"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

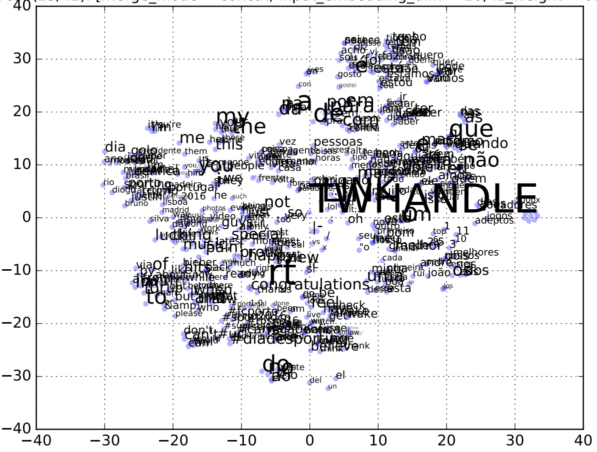
merge-mode=**merge**, input-emb-dim=**8**, l1-weight=0.00

Grid Test (22/42): [merge_mode = concat; input_embedding_dim = 8; l1_weight = 0.000000;



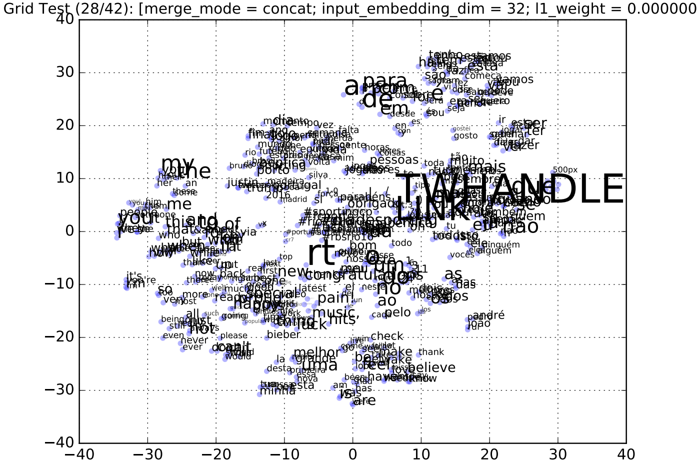
"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

Grid Test (25/42): [merge_mode = concat; input_embedding_dim = 16; l1_weight = 0.000000



input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim":
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

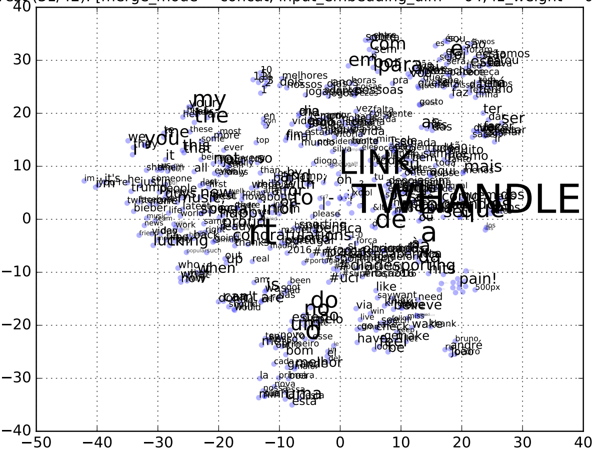
merge-mode=merge, input-emb-dim=**32**, l1-weight=0.00



input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim":
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 10

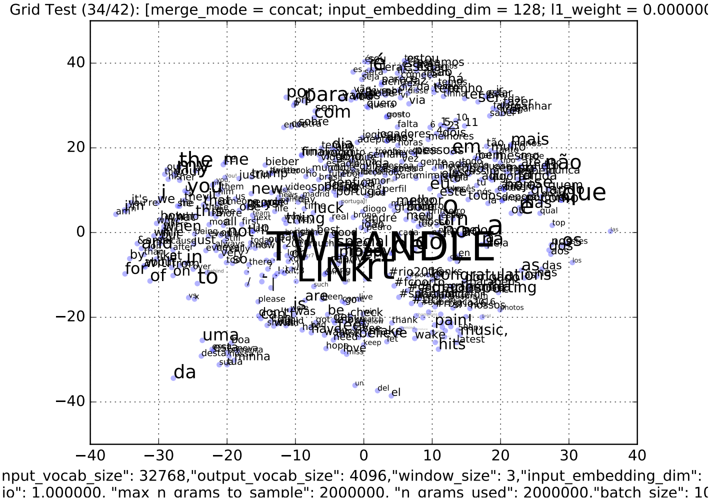
merge-mode=merge, input-emb-dim=**64**, l1-weight=0.00

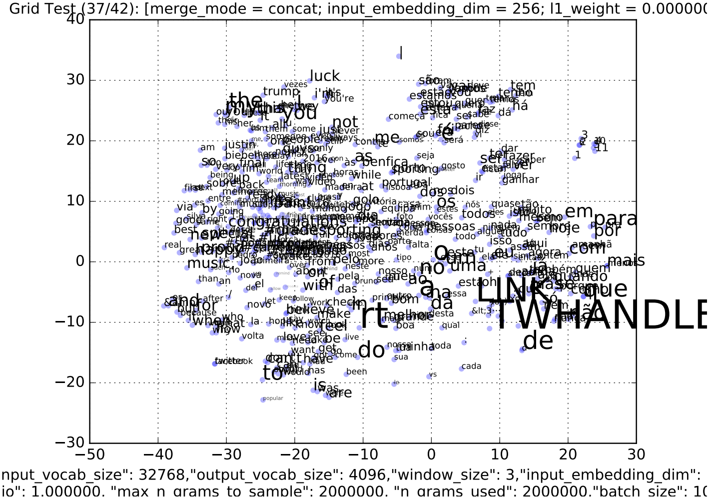
Grid Test (31/42): [merge_mode = concat; input_embedding_dim = 64; l1_weight = 0.000000

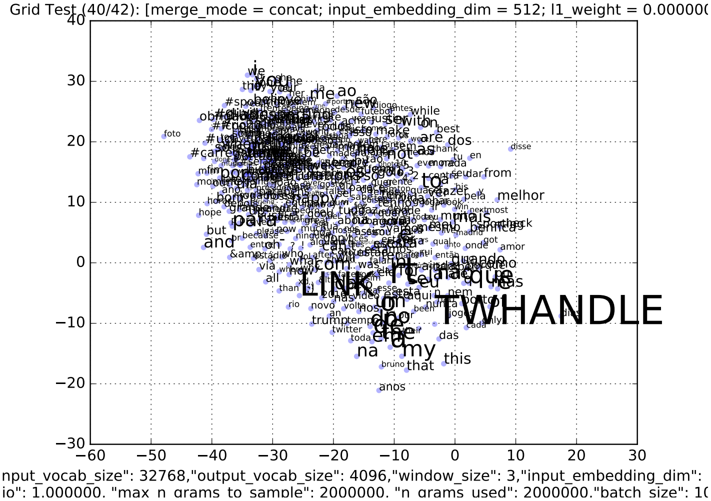


input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim":
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

merge-mode=merge, input-emb-dim=**128**, l1-weight=0.00

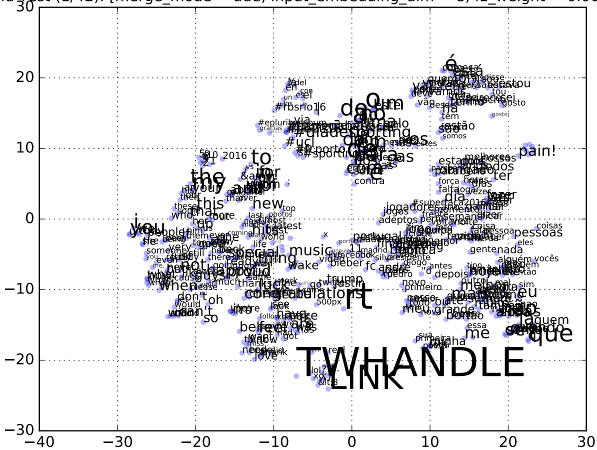



Grid Test (37/42): [merge_mode = concat; input_embedding_dim = 256; l1_weight = 0.000000
40
30
20
10
0
−10
−20
−30
−50
−40
−30
−20
−10
0
10
20
30
LINK
TWHANDLE
rt
de
do
to
que
para
em
com
mais
luck
not
congratulations
nput_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim":
io": 1.000000. "max_n_grams_to_sample": 2000000. "n_grams_used": 2000000."batch_size": 1(


Grid Test (40/42): [merge_mode = concat; input_embedding_dim = 512; l1_weight = 0.00000(
40
30
20
10
0
−10
−20
−30
−60
−50
−40
−30
−20
−10
0
10
20
30
TWHANDLE
LINK
nput_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim":
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

merge-mode=**add**, input-emb-dim=8, l1-weight=0.0

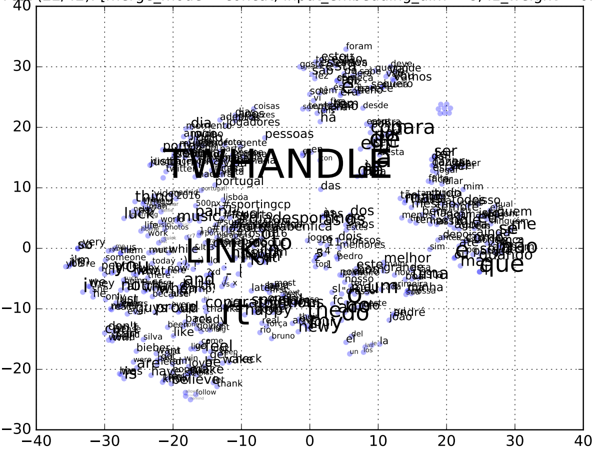
Grid Test (1/42): [merge_mode = add; input_embedding_dim = 8; l1_weight = 0.000000;



{"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

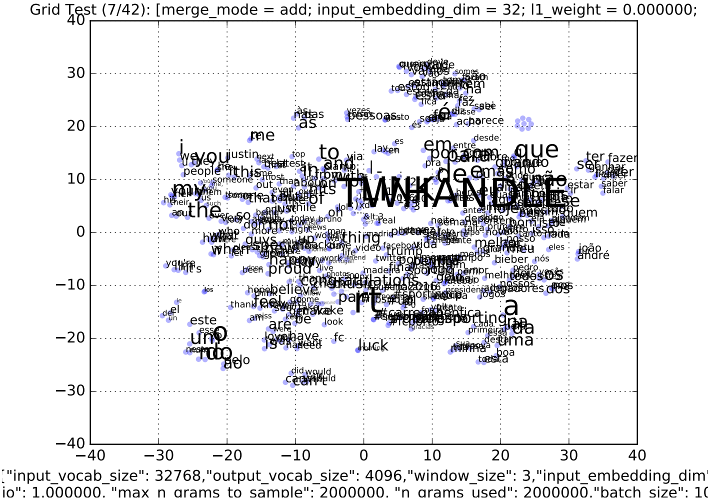
merge-mode=**concat**, input-emb-dim=8, l1-weight=0.0

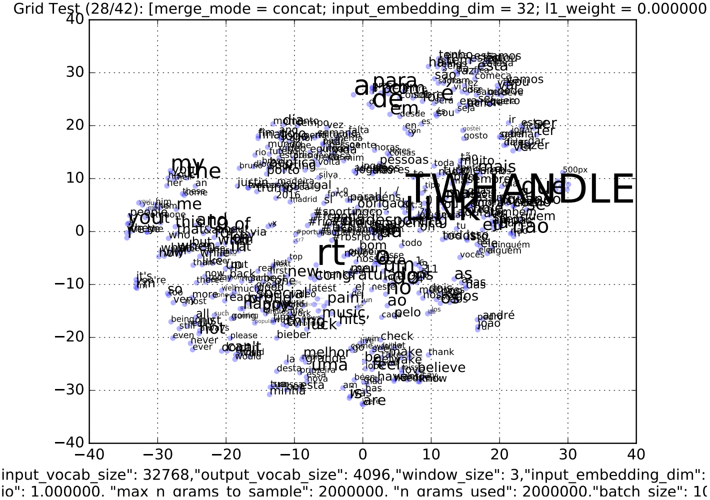
Grid Test (22/42): [merge_mode = concat; input_embedding_dim = 8; l1_weight = 0.000000;



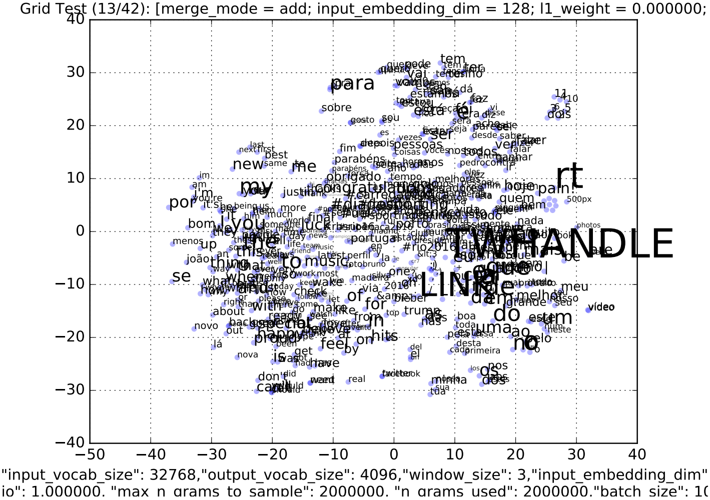
"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000. "max_n_grams_to_sample": 2000000. "n_grams_used": 2000000."batch_size": 1(

merge-mode=**add**, input-emb-dim=**32**, l1-weight=0.0



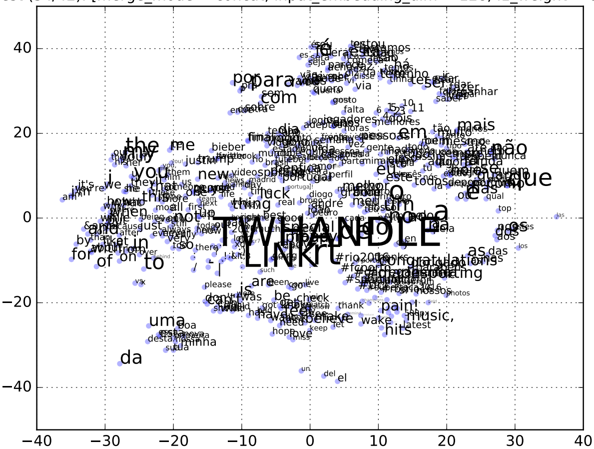

Grid Test (28/42): [merge_mode = concat; input_embedding_dim = 32; l1_weight = 0.000000
40
30
20
10
0
−10
−20
−30
−40
−40 −30 −20 −10 0 10 20 30 40
TWHANDLE
LINK
rt
input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim":
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

merge-mode=**add**, input-emb-dim=**128**, l1-weight=0.0



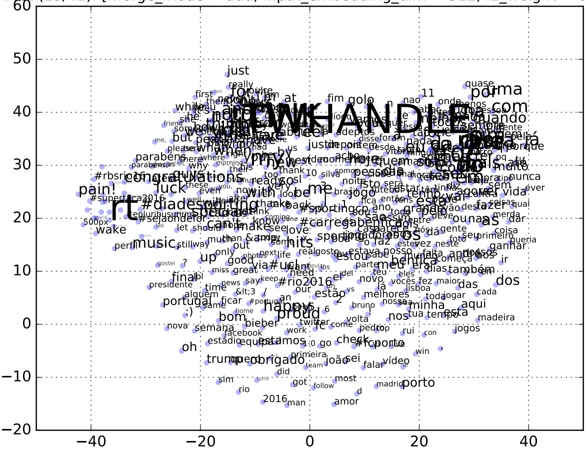
merge-mode=**concat**, input-emb-dim=128, l1-weight=0.0

Grid Test (34/42): [merge_mode = concat; input_embedding_dim = 128; l1_weight = 0.00000(

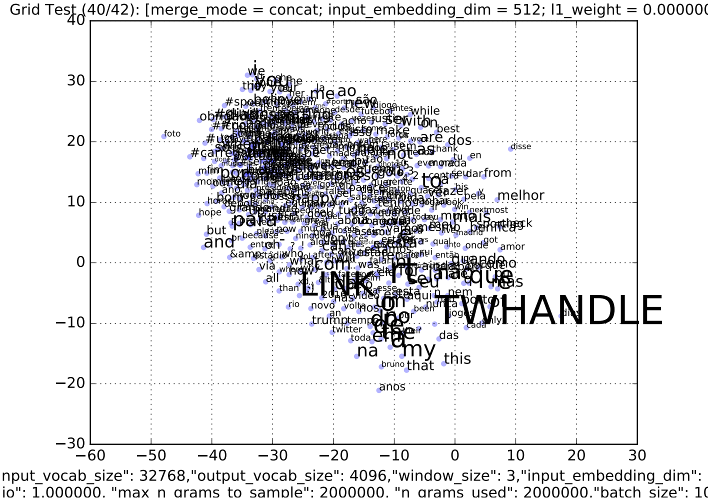


nput_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim":
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

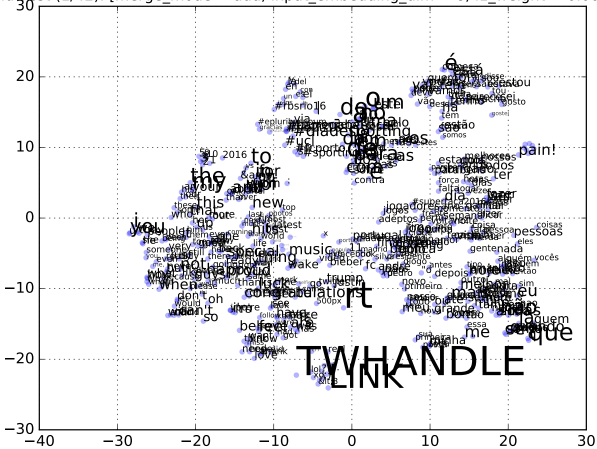
Grid Test (19/42): [merge_mode = add; input_embedding_dim = 512; l1_weight = 0.000000;



"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(


Grid Test (40/42): [merge_mode = concat; input_embedding_dim = 512; l1_weight = 0.00000(
40
30
20
10
0
−10
−20
−30
−60
−50
−40
−30
−20
−10
0
10
20
30
TWHANDLE
LINK
nput_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim":
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

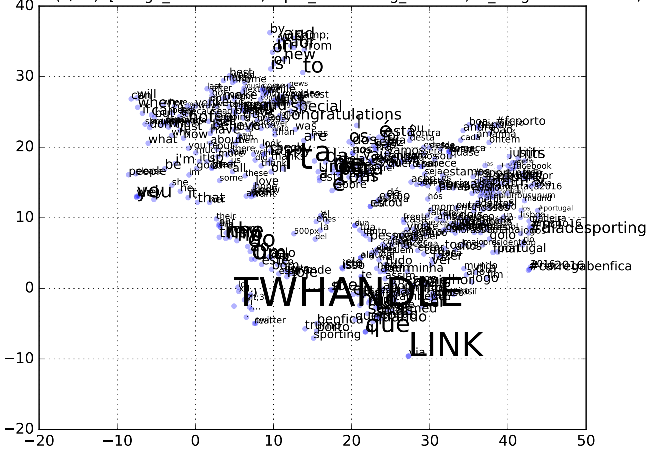
Grid Test (1/42): [merge_mode = add; input_embedding_dim = 8; l1_weight = 0.000000;



{"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 10

merge-mode=add, input-emb-dim=8, l1-weight=**0.0001**

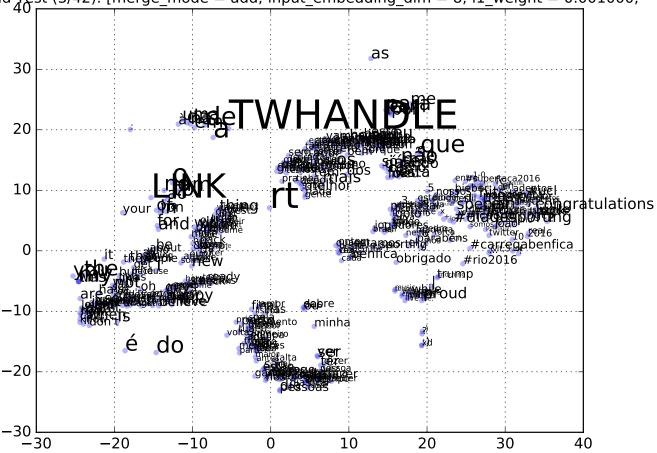
Grid Test (2/42): [merge_mode = add; input_embedding_dim = 8; l1_weight = 0.000100;



{"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 10

merge-mode=add, input-emb-dim=8, l1-weight=**0.001**

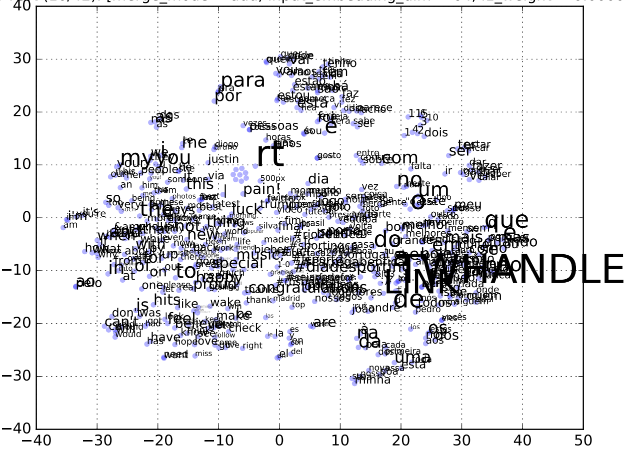
Grid Test (3/42): [merge_mode = add; input_embedding_dim = 8; l1_weight = 0.001000;



{"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

merge-mode=add, input-emb-dim=**64**, l1-weight=**0.00**

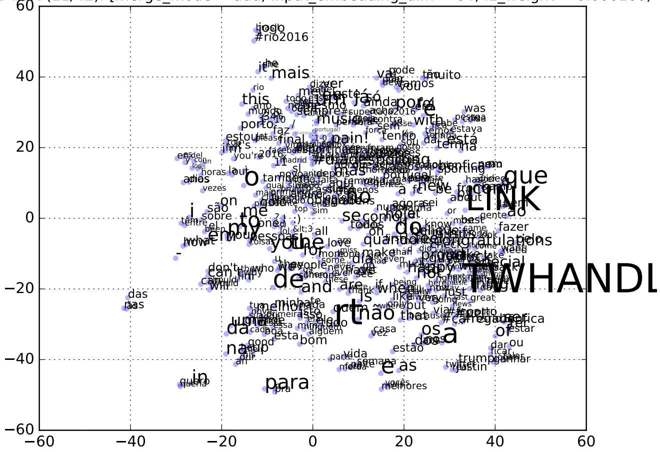
Grid Test (10/42): [merge_mode = add; input_embedding_dim = 64; l1_weight = 0.000000;



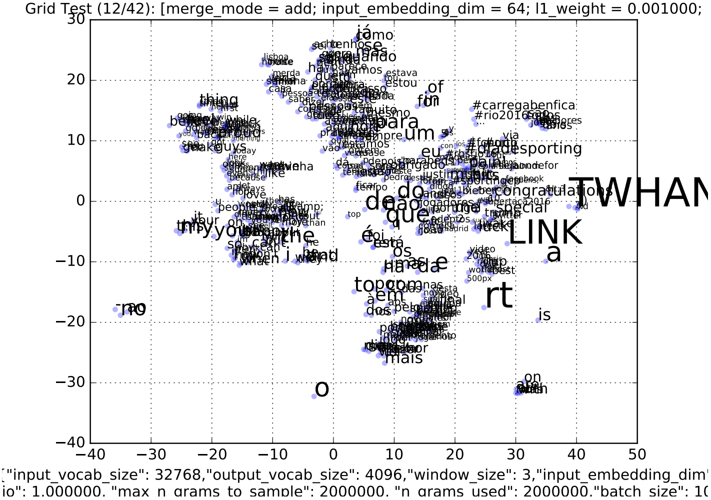
["input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

merge-mode=add, input-emb-dim=64, l1-weight=**0.0001**

Grid Test (11/42): [merge_mode = add; input_embedding_dim = 64; l1_weight = 0.000100;

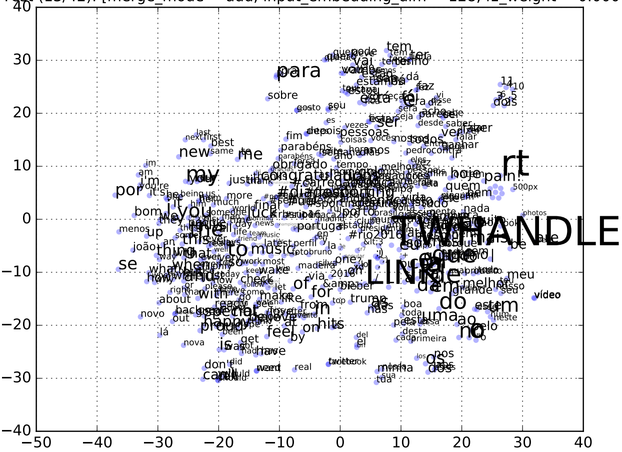


["input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim'
io": 1.000000. "max_n_grams_to_sample": 2000000. "n_grams_used": 2000000."batch_size": 1(


Grid Test (12/42): [merge_mode = add; input_embedding_dim = 64; l1_weight = 0.001000;
TWHAN
LINK
rt
a
["input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 10

merge-mode=add, input-emb-dim=**128**, l1-weight=**0.00**

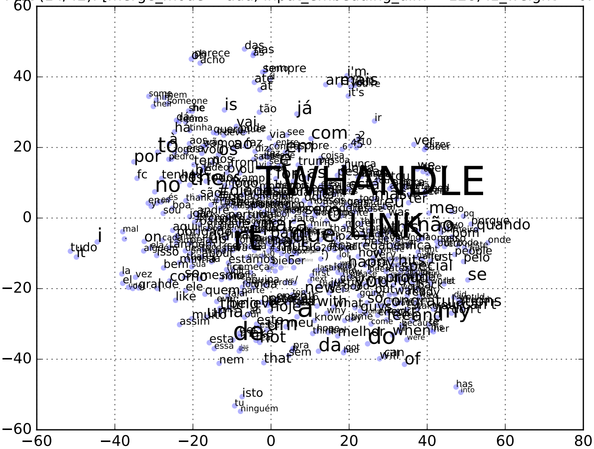
Grid Test (13/42): [merge_mode = add; input_embedding_dim = 128; l1_weight = 0.000000;



"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 1(

merge-mode=add, input-emb-dim=128, l1-weight=**0.0001**

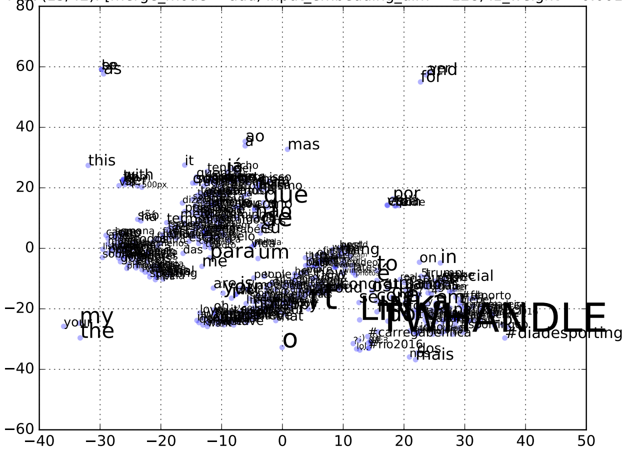
Grid Test (14/42): [merge_mode = add; input_embedding_dim = 128; l1_weight = 0.000100;



"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 10

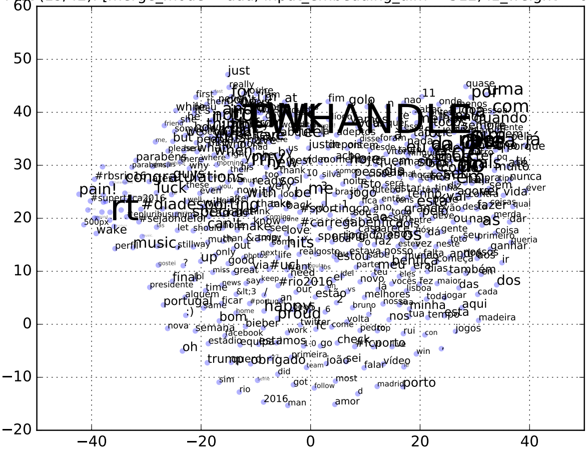
merge-mode=add, input-emb-dim=128, l1-weight=**0.001**

Grid Test (15/42): [merge_mode = add; input_embedding_dim = 128; l1_weight = 0.001000;



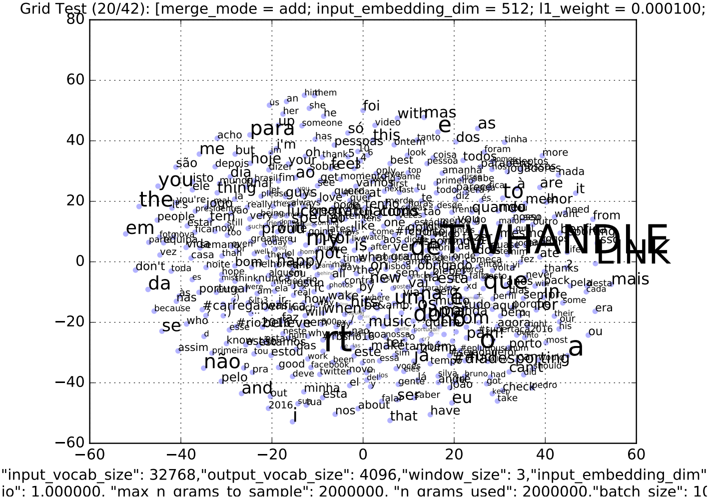
"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 10

Grid Test (19/42): [merge_mode = add; input_embedding_dim = 512; l1_weight = 0.000000;

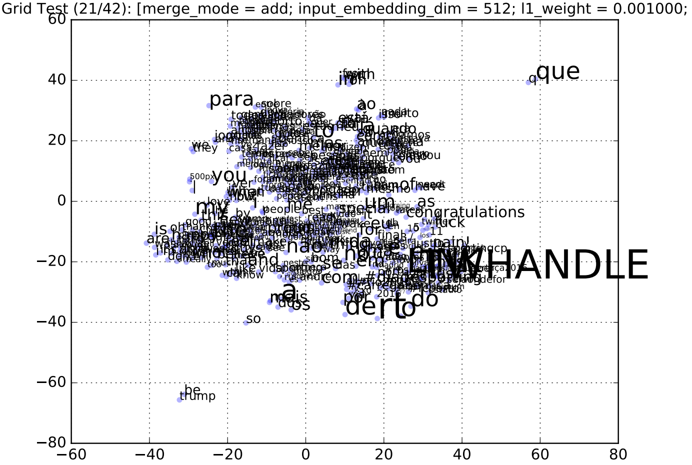


"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000. "max_n_grams_to_sample": 2000000. "n_grams_used": 2000000."batch_size": 1(

merge-mode=add, input-emb-dim=512, l1-weight=**0.0001**



merge-mode=add, input-emb-dim=512, l1-weight=**0.001**



"input_vocab_size": 32768,"output_vocab_size": 4096,"window_size": 3,"input_embedding_dim"
io": 1.000000, "max_n_grams_to_sample": 2000000, "n_grams_used": 2000000,"batch_size": 10

What's next?

What's next?

- More systematic evaluation:
  - visual inspection helps but it is not enough
- Producing a Gold Standard
  - manual compilation (e.g. "months", "soccer clubs")
  - extracting lists from Wikipedia
  - aggregation of other existing evaluation resources
- Main Challenges:
  - lack coverage (specially for UGC)
  - lack of agreement on the level of "fidelity" / task:
    - same "level" class? ("Porto", "Lisboa")
    - is-a? ("London", "city")
    - synonyms / equivalence ("pq", "porque")
    - how to deal with antonyms
- How do design embeddings to separate this task

What's next?

- These are not just simple details:
  - they are fundamentally related with the semantic information the models are able to capture!
- We need to design:
  - more models,
  - X,Y task

  that capture **different** semantic **by design**
- Lot's of room for experimentation
  - exploration / "learning" of alternative architectures
  - multi-task learning to incorporate more semantic facets in the embedding space
  - working with other media types (e.g. voice may allow co-encode aspect related to sentiment)

And now it's time for...

Questions!

Thank you!

luis.sarmento@gmail.com

Connect with me on LinkedIn
https://www.linkedin.com/in/luissarmento/