Integrating and Evaluating Neural Word Embeddings in Information Retrieval

Guido Zuccon[1], Bevan Koopman[2,1], Peter Bruza[1], Leif Azzopardi[3]

[1] Queensland University of Technology (QUT), Brisbane, Australia
[2] Australian E-Health Research Centre, CSIRO, Brisbane, Australia
[3] University of Glasgow, Glasgow, United Kingdom

Word Embeddings (and IR)

- recent increased interest in developing word embeddings (WE) and applying them to language tasks
- spark of usage of word embeddings in IR
- the **hypothesis** of our work is that **Word Embeddings** can be **exploited to search** documents - with W.E. addressing the **semantic gap** problem
    - to suggest/interpret queries [some previous work, e.g. Blanco et al, WSDM'15]
    - to **retrieve documents** [this work]

What are word embeddings?

- **word embedding** = a (real-valued) **vectorial representation of a word** (of smaller dimensionality than original vocabulary)

Information $\longrightarrow$ $[x_1, x_2, \ldots x_i \ldots x_n]$

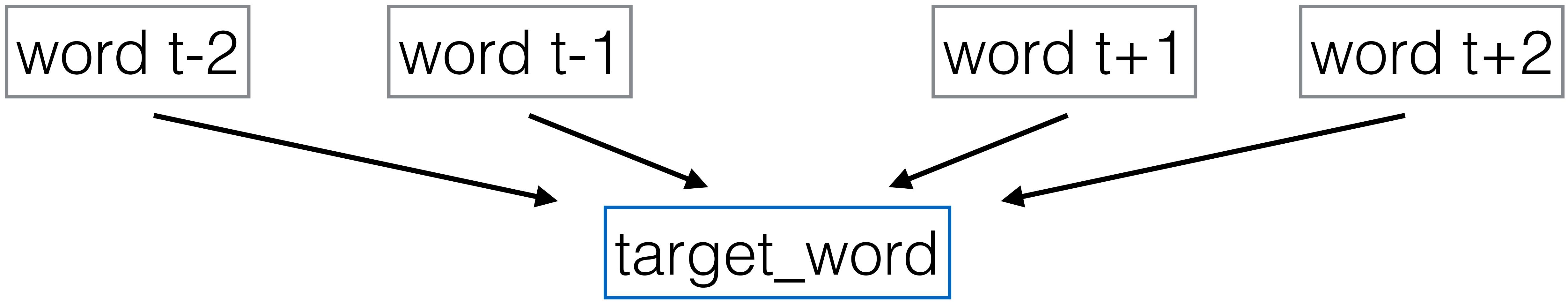
Retrieval $\longrightarrow$ $[y_1, y_2, \ldots y_i \ldots y_n]$

- mappings from words to vectors are learnt through optimisation of objective function

- focus on Neural LM: representation is learnt by training a neural network LM
  (a very simple one in this paper)

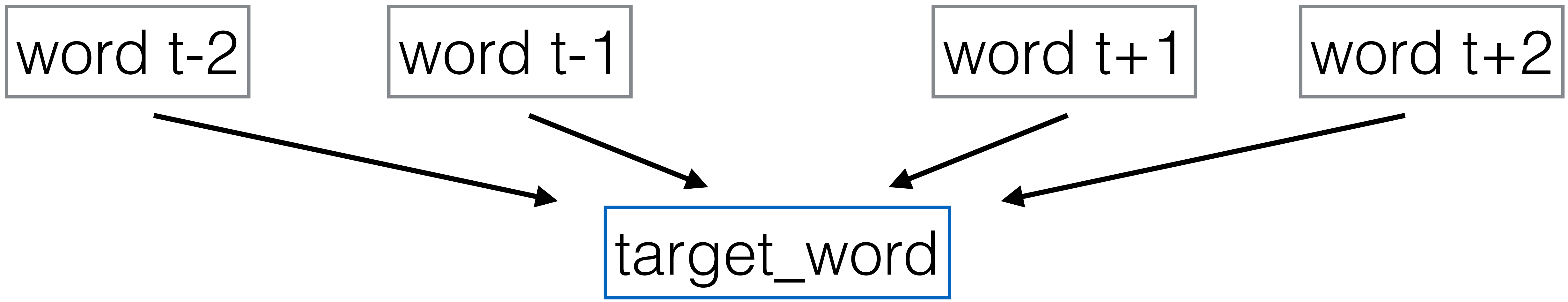
Skipgrams and CBOW

CBOW:

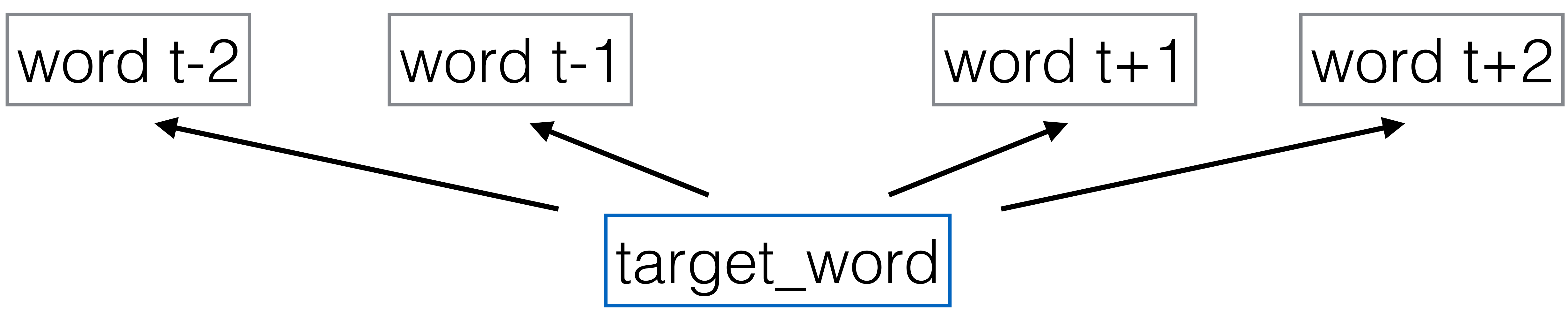


Skipgrams and CBOW

CBOW:



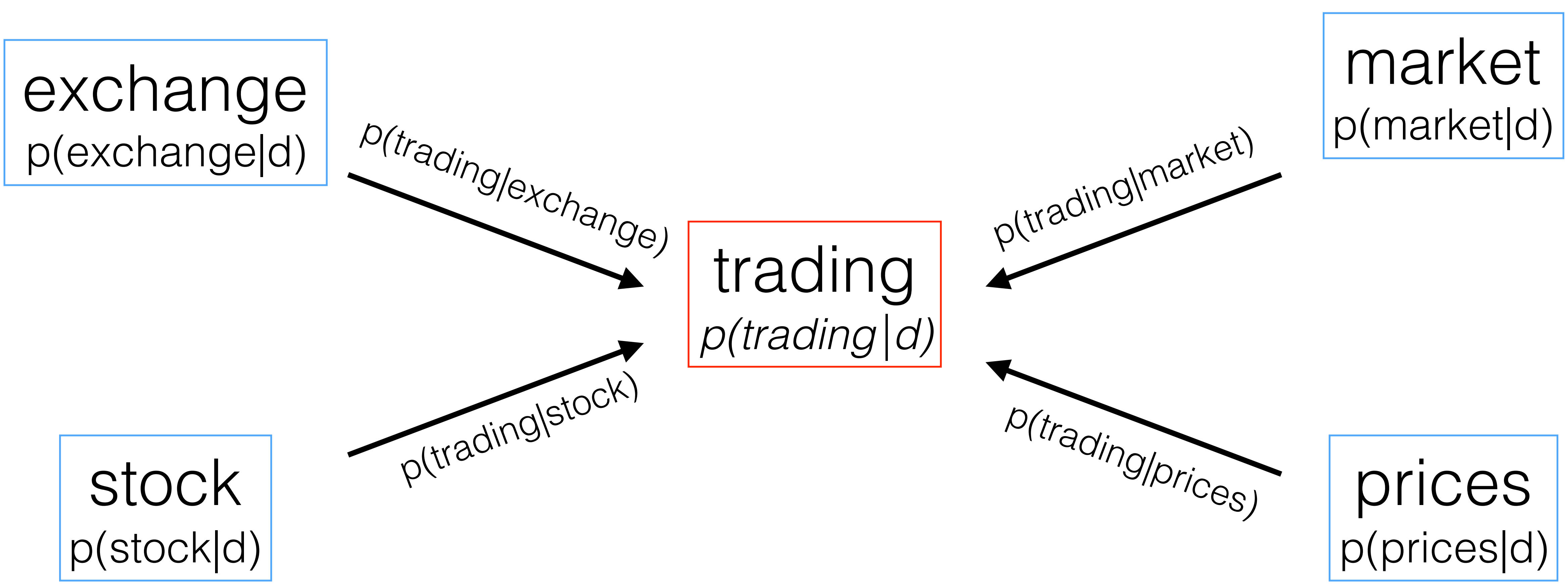
Skipgram:



Translation Language Models

compute *p(w|d)* in function of all words *u* that translate to *w*, the likelihood of the translation, and their likelihood in the document, *p(u|d)*

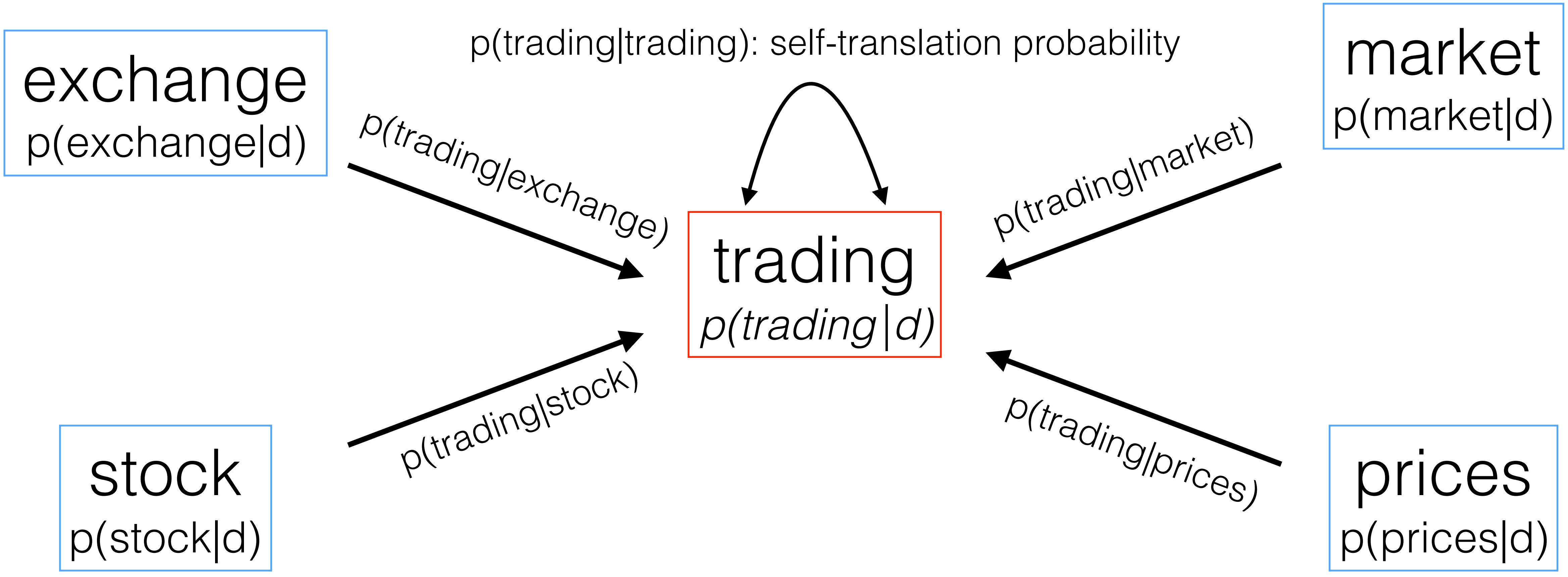
$$p_t(w|d) = \sum_{u \in d} p_t(w|u)p(u|d)$$



Translation Language Models

compute *p(w|d)* in function of all words *u* that translate to *w*, the likelihood of the translation, and their likelihood in the document, *p(u|d)*

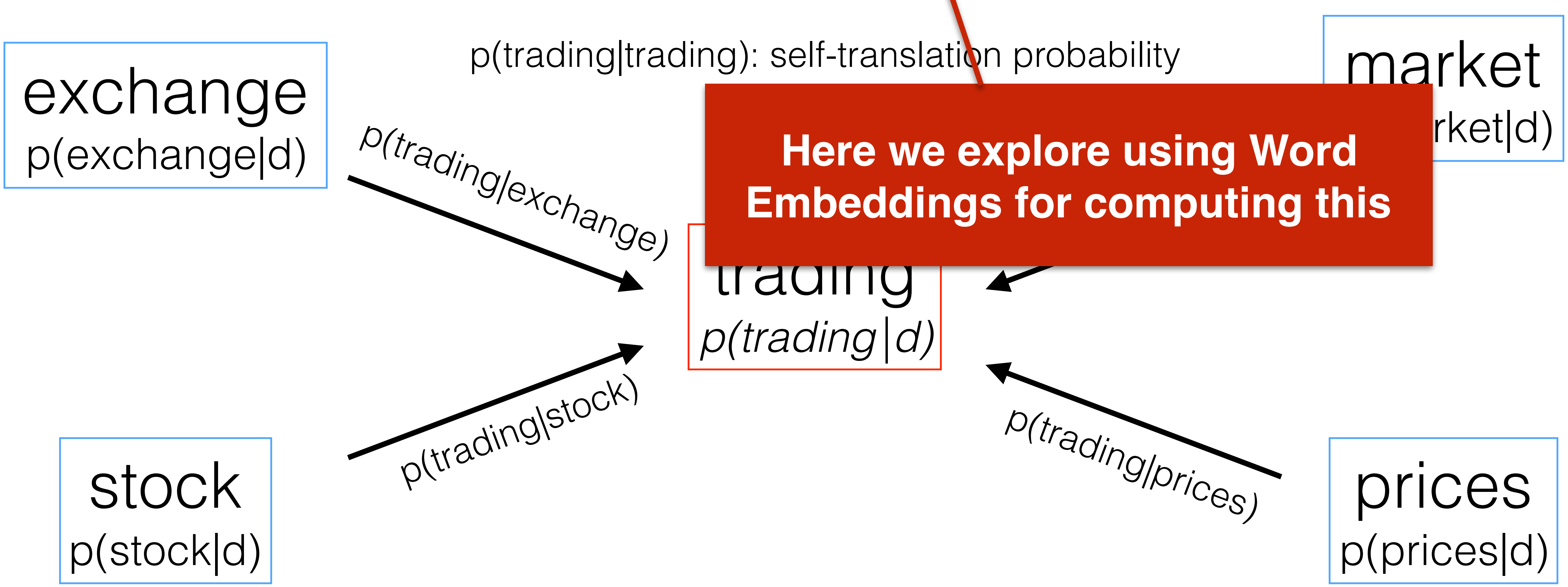
$$p_t(w|d) = \sum_{u \in d} p_t(w|u) p(u|d)$$



Translation Language Models

compute *p(w|d)* in function of all words *u* that translate to *w*, the likelihood of the translation, and their likelihood in the document, *p(u|d)*

$$p_t(w|d) = \sum_{u \in d} \boxed{p_t(w|u)}\, p(u|d)$$



NTLM: Integrating Word Embeddings into TLM

- a word is represented by a vector
- can compute translation probabilities by measuring cosine similarity between word vectors

$$p_{cos}(w|u) = \frac{cos(w,u)}{\sum_{u' \in V} cos(w,u')}$$

*Note: since vectors are positive valued, *0 <= cos(w|u) <= 1*

How is NTLM different?

TLM-MI

w = insider		w = trading	
u	p(w|u)	u	p(w|u)
insider	0.094	trading	0.050
trading	0.023	exchange	0.016
securities	0.023	stock	0.014
fraud	0.015	market	0.013
drexel	0.013	prices	0.012
burnham	0.013	traders	0.009
lambert	0.012	index	0.009
stock	0.012	shares	0.009
wall	0.011	stocks	0.009
commission	0.010	dow	0.008

NTLM

w = insider		w = trading	
u	p(w|u)	u	p(w|u)
insider	0.169	trading	0.164
fraud	0.102	traders	0.103
drexel	0.099	futures	0.099
securities	0.096	stock	0.097
racketeering	0.093	exchang	0.094
bribery	0.091	market	0.093
scandal	0.089	brokers	0.088
criminal	0.088	dealings	0.088
burnham	0.088	mercantil	0.088
scheme	0.085	securities	0.086

How is NTLM different?

TLM-MI

w = insider		**w = trading**	
u	p(w|u)	u	p(w|u)
insider	0.094	**trading**	0.050
trading	0.023	exchange	0.016
securities	0.023	stock	0.014
fraud	0.015	market	0.013
drexel	0.013	prices	0.012
burnham	0.013	traders	0.009
lambert	0.012	index	0.009
stock	0.012	shares	0.009
wall	0.011	stocks	0.009
commission	0.010	dow	0.008

NTLM

w = insider		**w = trading**	
u	p(w|u)	u	p(w|u)
insider	0.169	**trading**	0.164
fraud	0.102	traders	0.103
drexel	0.099	futures	0.099
securities	0.096	stock	0.097
racketeering	0.093	exchang	0.094
bribery	0.091	market	0.093
scandal	0.089	brokers	0.088
criminal	0.088	dealings	0.088
burnham	0.088	mercantil	0.088
scheme	0.085	securities	0.086

NTLM guarantees self-translation prob. are highest translations

TLM-MI doesn't; Axiomatic analysis found this is a requirement

[Karimzadehgan&Zhai, ECIR'12]

How is NTLM different?

TLM-MI

w = insider		w = trading	
u	p(w|u)	u	p(w|u)
insider	**0.094**	trading	0.050
trading	0.023	exchange	0.016
securities	0.023	stock	0.014
fraud	0.015	market	0.013
drexel	0.013	prices	0.012
burnham	0.013	traders	0.009
lambert	0.012	index	0.009
stock	0.012	shares	0.009
wall	0.011	stocks	0.009
commission	0.010	dow	0.008

NTLM

w = insider		w = trading	
u	p(w|u)	u	p(w|u)
insider	**0.169**	trading	0.164
fraud	0.102	traders	0.103
drexel	0.099	futures	0.099
securities	0.096	stock	0.097
racketeering	0.093	exchang	0.094
bribery	0.091	market	0.093
scandal	0.089	brokers	0.088
criminal	0.088	dealings	0.088
burnham	0.088	mercantil	0.088
scheme	0.085	securities	0.086

NTLM has relatively lower self-translation probabilities

Translations may have more effect than for TLM-MI

How is NTLM different?

TLM-MI

w = insider		**w = trading**	
u	p(w|u)	u	p(w|u)
insider	0.094	trading	0.050
trading	0.023	exchange	0.016
securities	0.023	stock	0.014
fraud	0.015	market	0.013
drexel	0.013	prices	0.012
burnham	0.013	**traders**	**0.009**
lambert	0.012	index	0.009
stock	0.012	shares	0.009
wall	0.011	stocks	0.009
commission	0.010	dow	0.008

NTLM

w = insider		**w = trading**	
u	p(w|u)	u	p(w|u)
insider	0.169	trading	0.164
fraud	0.102	**traders**	**0.103**
drexel	0.099	futures	0.099
securities	0.096	stock	0.097
racketeering	0.093	exchang	0.094
bribery	0.091	market	0.093
scandal	0.089	brokers	0.088
criminal	0.088	dealings	0.088
burnham	0.088	mercantil	0.088
scheme	0.085	securities	0.086

NTLM gives higher prob. to syntactic variations of query

Performs an implicit stemming

Empirical Validation: Research Questions

- **RQ1:** Do NTLM provide translation probabilities that lead to **improved retrieval effectiveness** (compared to other TLMs)
- **RQ2:** How **sensitive** is NTLM to differences in word embedding constructions?
(embedding dimensionality, window size)
- **RQ3:** Does the choice of **training corpus** for embeddings impact NTLM effectiveness?
(can we try on a common, general corpus?)

Experiment Settings

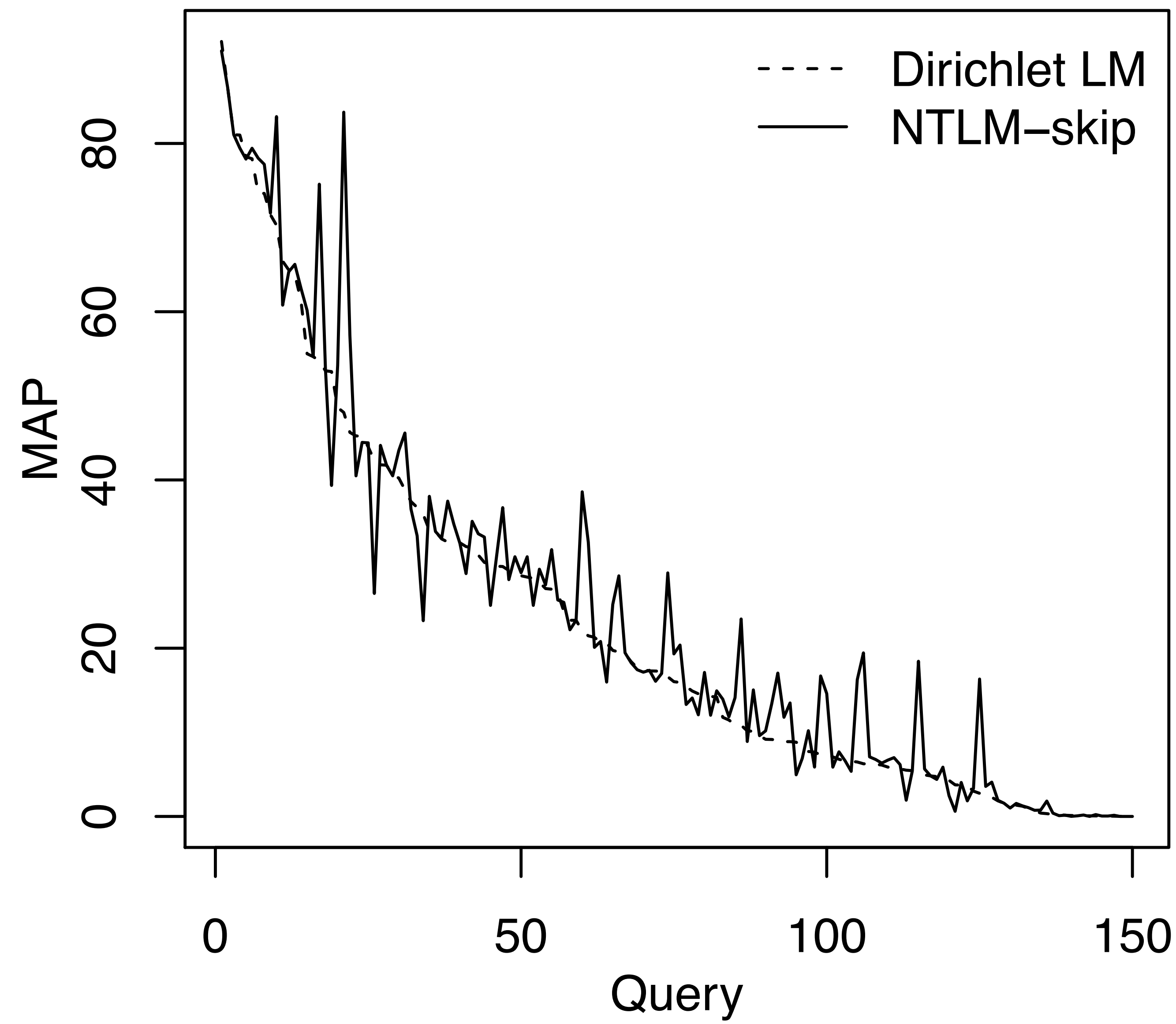
- Evaluate on **ad-hoc, medical** collections
  - adhoc were used in previous evaluation of TLM (AP88-89, WSJ87-92)
  - medical is characterised by semantic gap problem (Medtrack)
- also use **DOTGOV**: TLM never evaluated on larger collections
  - mainly due to processing involved to compute MI

Experiment Settings

- **Baseline**: Dirichlet LM , tuned per-collection and per-topic set wrt MAP (bpref)
    - same baseline and tuning regime as previous TLM work
- **Benchmark**: Mutual Information (MI) based TLM [Karimzadehgan&Zhai, SIGIR'10]
    - 3 versions: standard, alpha-MI-TLM, s-MI-TLM; tuned per-collection and per-topic set
- **NTLM**: control dimensionality, window size, training corpus
    - fix other parameters: negative sampling (20), subsampling, iterations (5)
- For all, only use top 10 translation terms

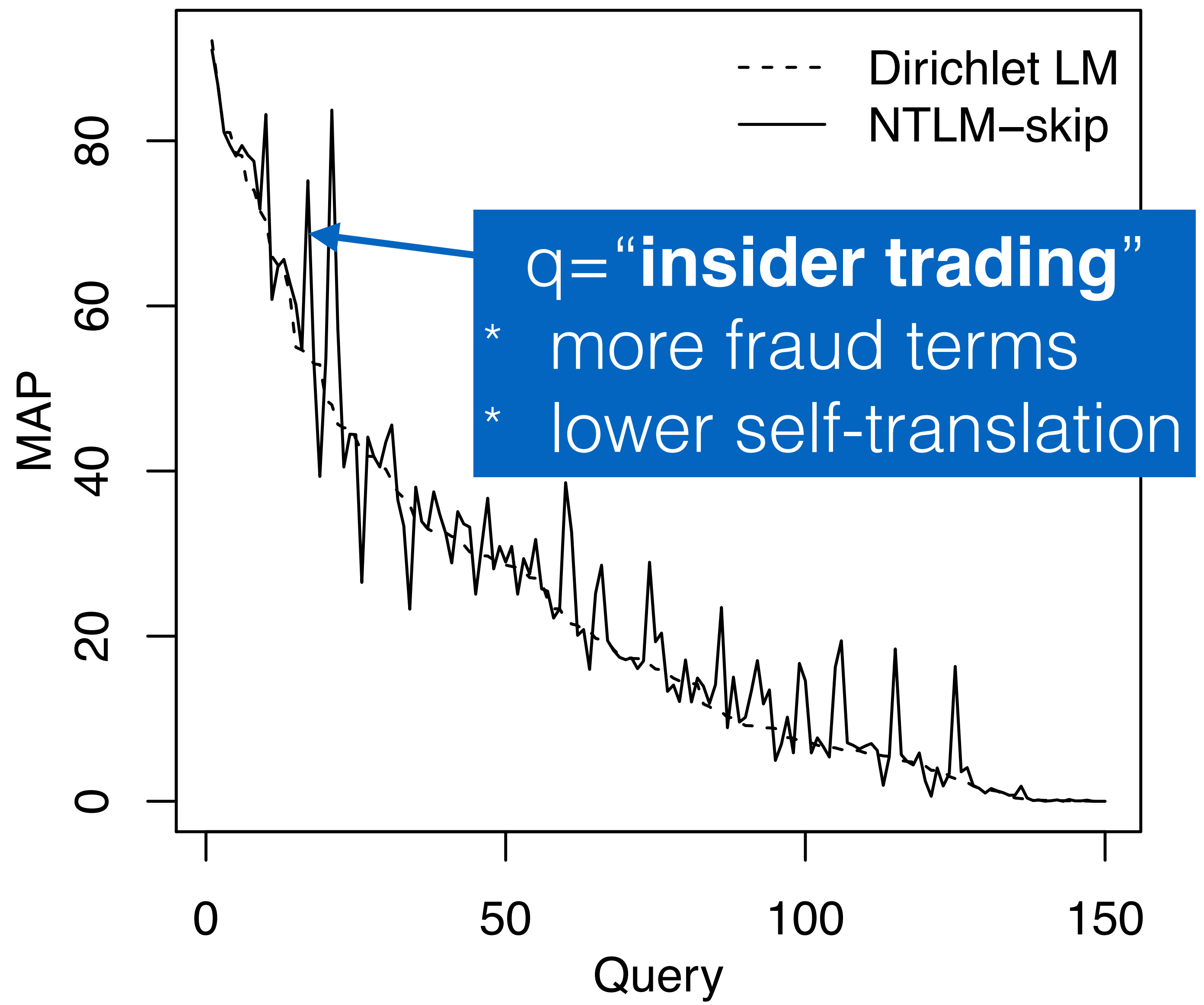
How does NTLM compare to LM & TLM?

- modest improvements on large number of queries



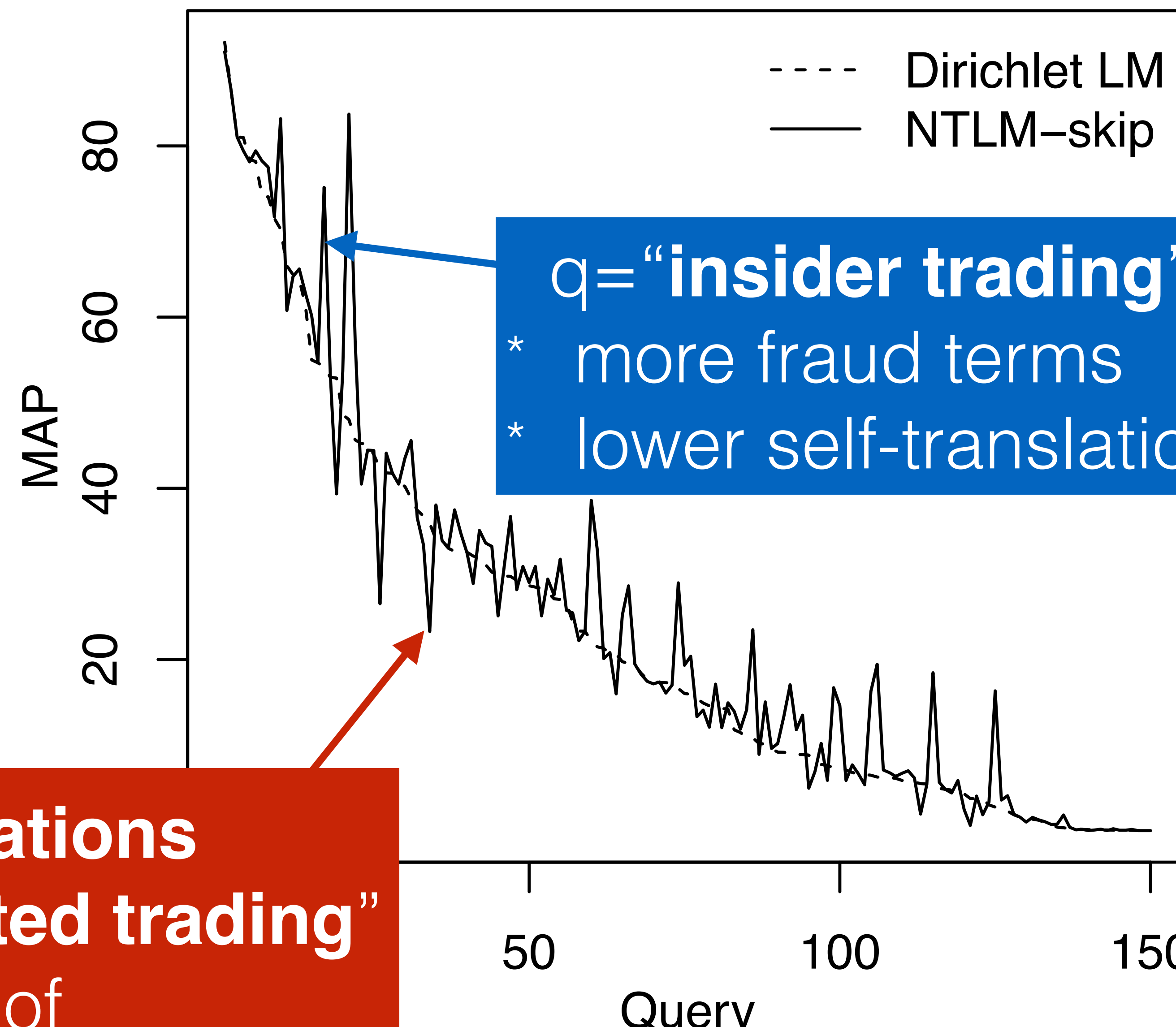
How does NTLM compare to LM & TLM?

- modest improvements on large number of queries



How does NTLM compare to LM & TLM?

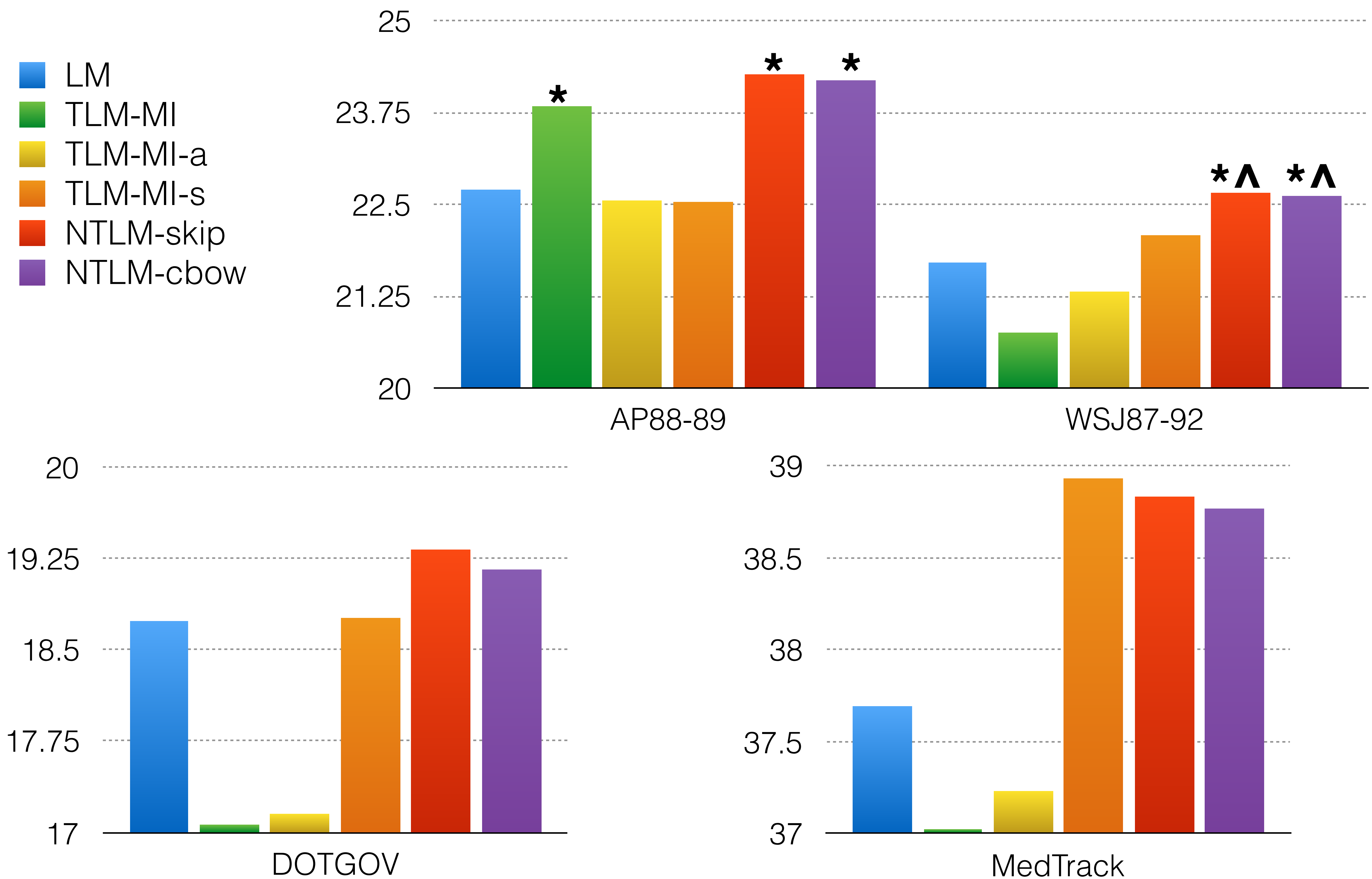
- modest improvements on large number of queries



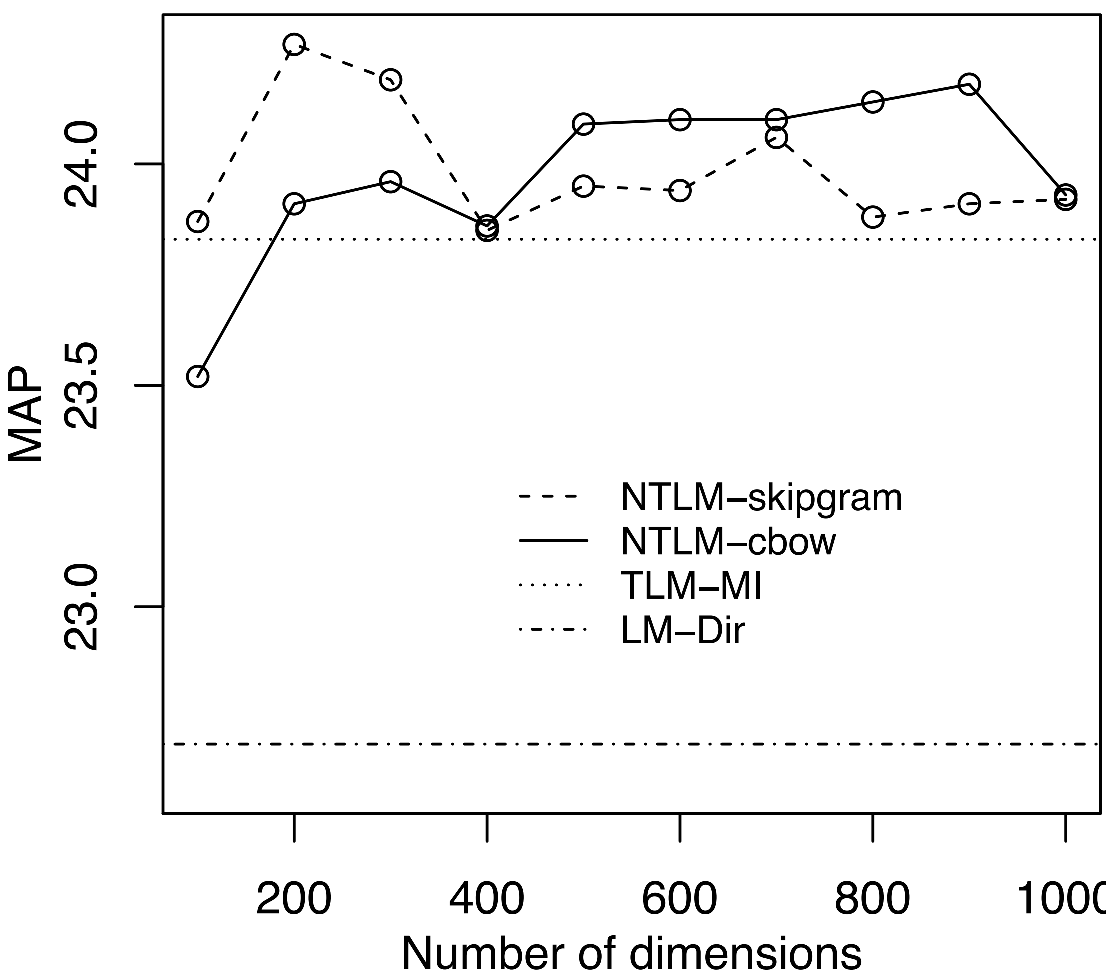
q="**insider trading**"
* more fraud terms
* lower self-translation

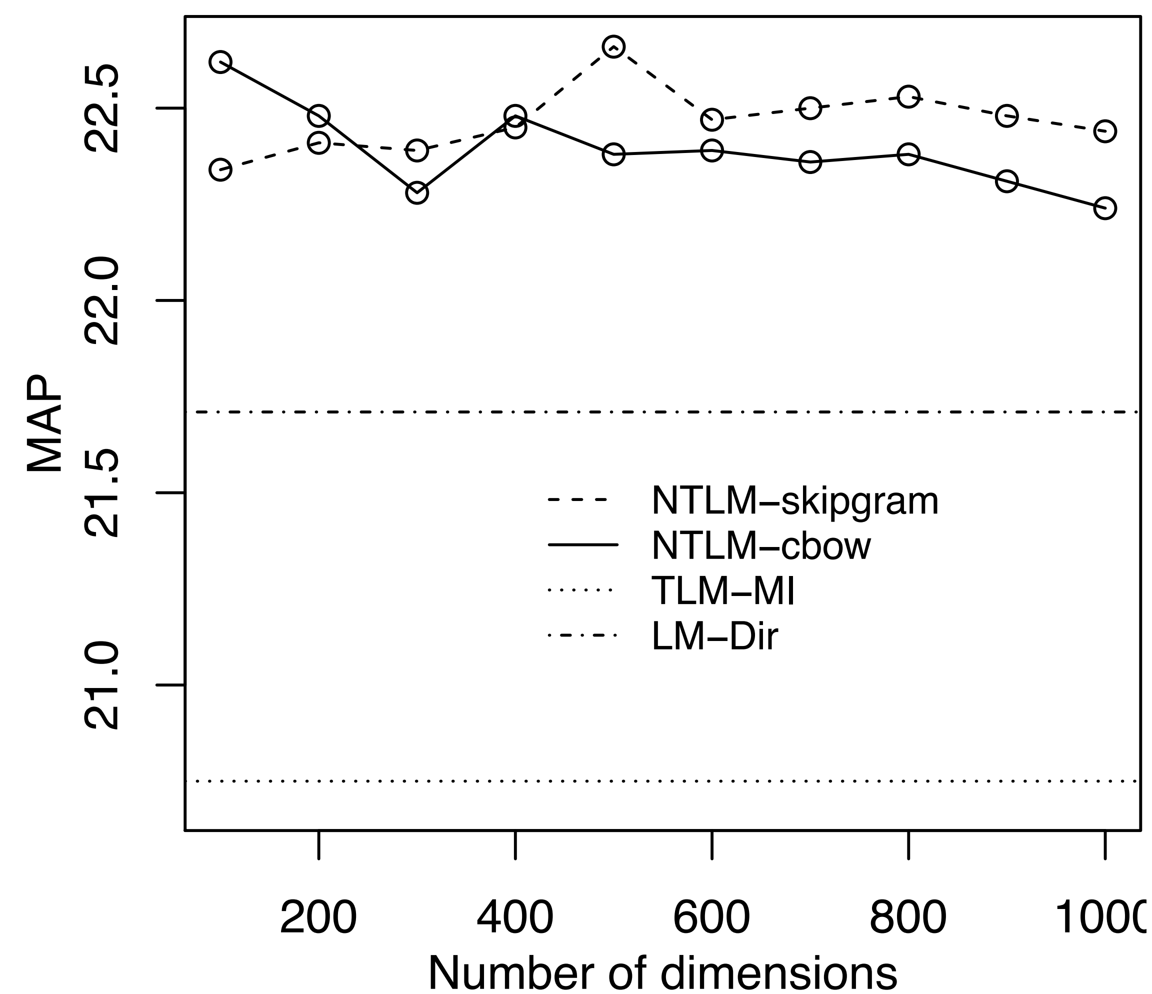
q="**stock market perturbations attributable to computer initiated trading**"
* "video", "chip" as translations of "computer"

How does NTLM compare to LM & TLM?



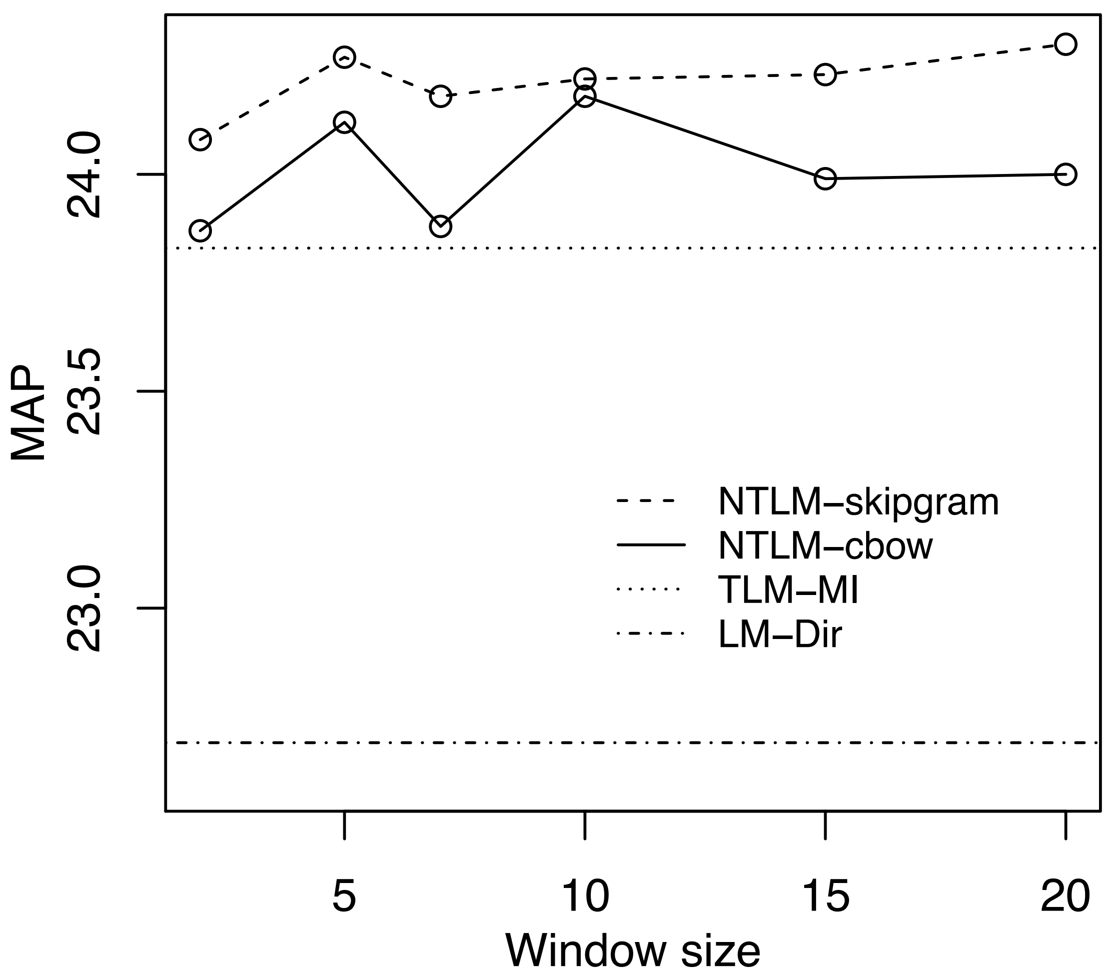
What is the Effect of Embedding Dimensionality?

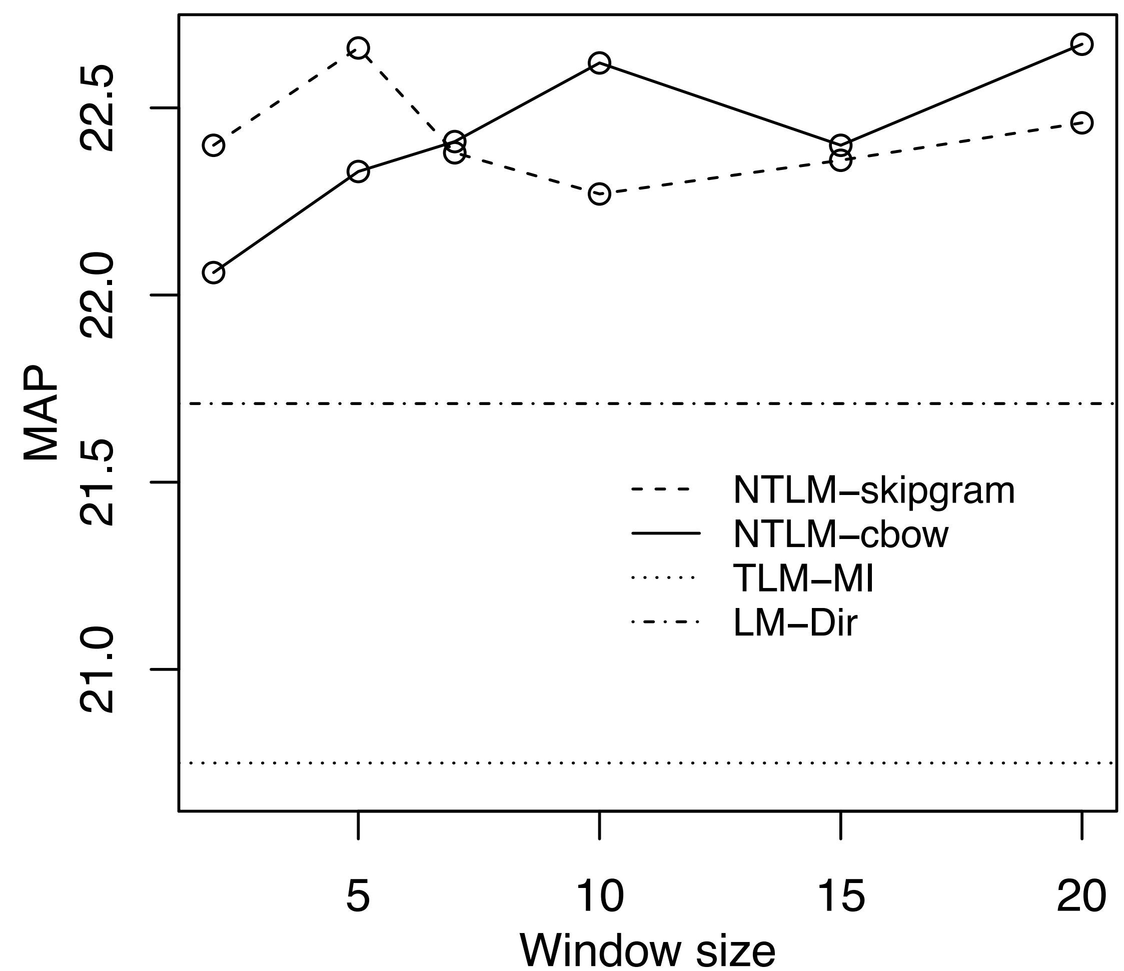




Choice in **dimension** size **not significantly** affect effectiveness

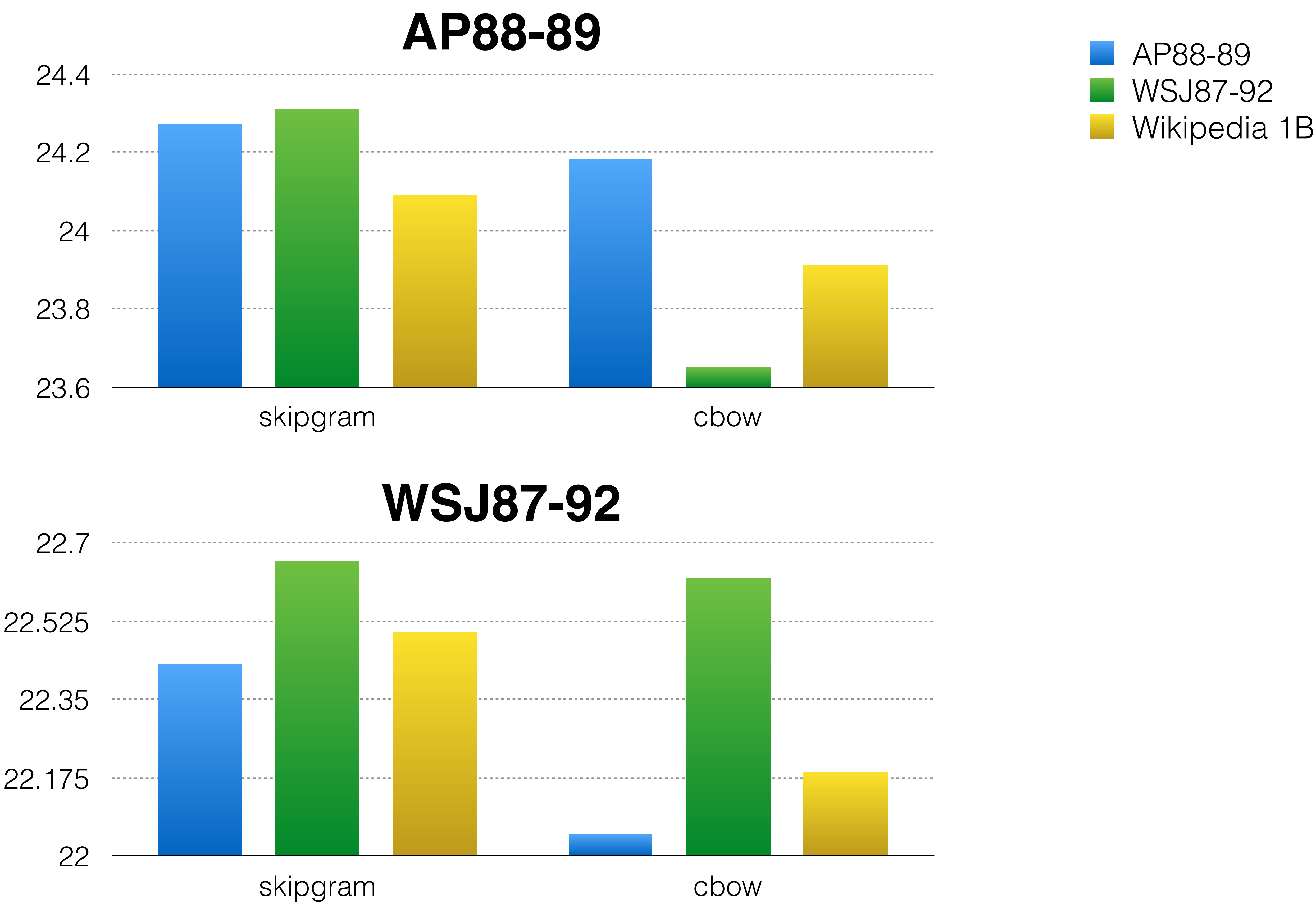
What is the Effect of Window Size?



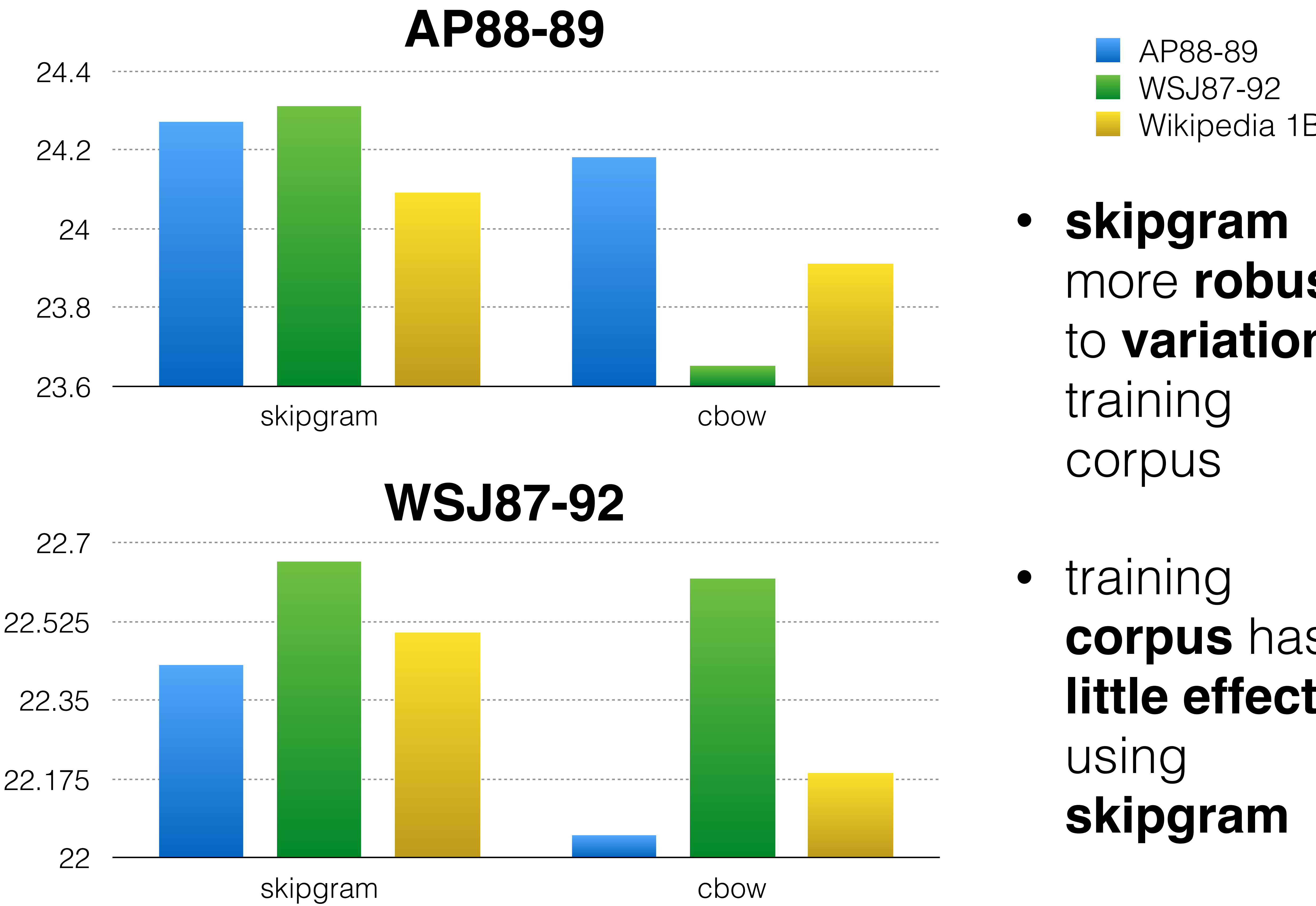


Choice in **window** size **not significantly** affect effectiveness

What is the Effect of Training Corpus?



What is the Effect of Training Corpus?



- **skipgram** more **robust** to **variation** of training corpus
- training **corpus** has **little effect** if using **skipgram**

Summary of Contributions

- **Theory**: integration of word embeddings in retrieval model
- **Empirical**: NTLM better than LM and similar to TLM on small corpora (but faster - not evaluated in this work)
- Understanding of **impact of embedding settings**:
    - **dimensionality**: no impact
    - **window size**: no impact
- Word **embeddings** could be **reused** on other data
- Code, results and word embeddings available at https://github.com/ielab/adcs2015-NTLM

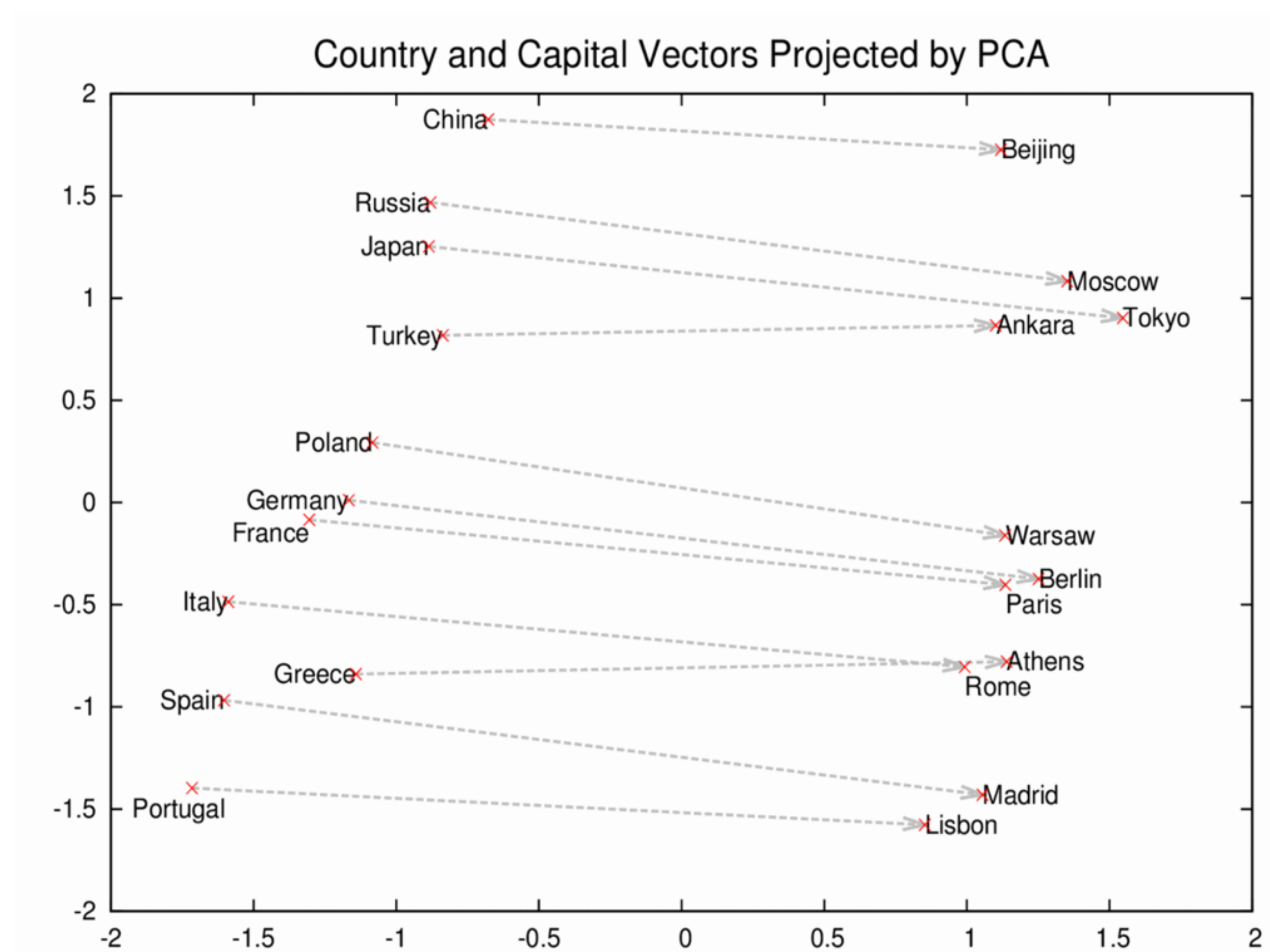
Outlook

- embeddings can be used across collections: word embeddings as a service

- word embeddings allow for reasoning via vector operations:
  - compositionality = vector sum
  - analogy:

$$\arg\max_{b^* \in V} \frac{\cos(b^*, b)\cos(b^*, a^*)}{\cos(b^*, a) + \varepsilon}$$

($\varepsilon = 0.001$ is used to prevent division by zero)



can we use word embeddings to define query operations?