From Boolean Towards Semantic Retrieval Models

Speakers : Arpan Gupta, Seinjuti Chatterjee

About us

Leading Machine Learning Platform For Ecommerce Search

120+Customers & Brands | 1200+ Global Websites | Over 1.5 Billion Interactions/Month

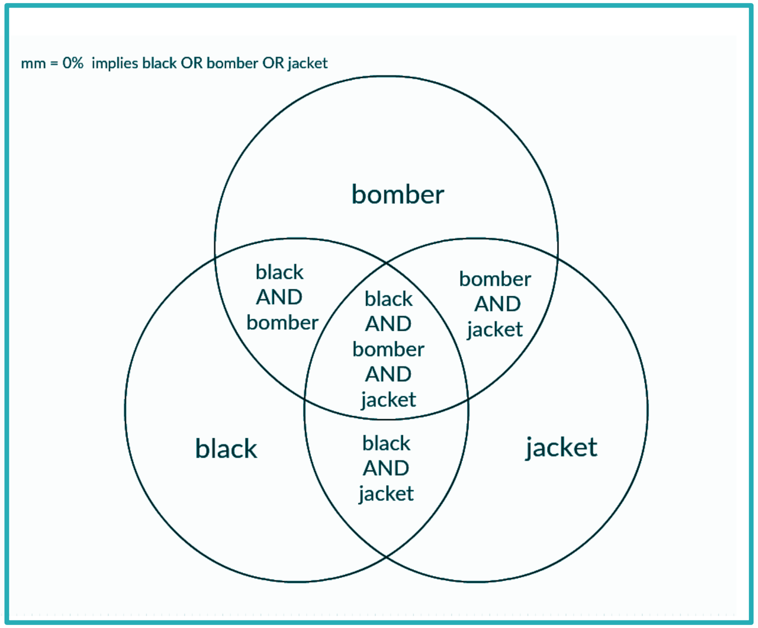
FASHION	HARDWARE & B2B	ELECTRONICS	GROCERY & PHARMA	HOME IMPROVEMENT	PLATFORMS
EXPRESS	ib supply	HSN	freshdirect	Ashley HOMESTORE	atg
rue21	US AUTOPARTS	SCOSCHE	HealthyPlanet	BED BATH & BEYOND	IBM WebSphere
U.S. PATRIOT TACTICAL	Parts Warehouse	sylvane	Himalaya HERBAL HEALTHCARE	mood	Magento Enterprise
burkes OUTLET	Freund CONTAINER & SUPPLY	ITRenewDirect	WebPT	KNOBBERY	hybris
deborah lippmann	sfcable.com	Cables.com	medical supply DEPOT	pepperfry	demandware

Unbxd – Product Discovery Platform

Built on Machine Learning and AI to drive better experiences, engagement, and ultimately drive conversions!

- Site Search
- Intelligent Storefront
- Product Recommendations

Boolean Retrieval



- Query = “**black bomber jacket**”
- Search Setting: Minimum match(MM)
  - **MM=100%** ⇔ match any term, **MM=0%** ⇔ match all terms
  - MM=66% ⇔ also matches "black bomber". Far from good
  - MM can’t specify importance of the terms.
- Better relevance ⇔ weighted retrieval(weighted query terms)

Semantic Retrieval

- Classic relevance measures
  - Precision = num relevant docs/num retrieved docs
  - Recall = num relevant docs retrieved/num actual relevant docs

- Better relevance → Semantic retrieval (Key idea of this talk)
  - ⇔ Identify MT(must have) tokens
    - Improves precision but may drop recall
  - ⇔ Augment MTs by synonyms (word sense disambiguated)
    - As disjunctive(OR)s of MTs for better recall

Must Have Tokens improve precision

MT : What noun best describes this product?

?

One classic-cool silhouette, two slick bomber jacket options. This reversible layer doubles your wardrobe by letting you switch from black to green to match tons of looks.

- Reversible bomber jacket
- Stand-up collar; Zip front
- Long sleeves; Pocket on left arm
- Hand pockets; Straight hem
- Polyester
- Machine wash
- Imported

Must Have Tokens improve precision

MT : What noun best describes this product?

Bomber Jacket

One classic-cool silhouette, two slick bomber jacket options. This reversible layer doubles your wardrobe by letting you switch from black to green to match tons of looks.

- Reversible bomber jacket
- Stand-up collar; Zip front
- Long sleeves; Pocket on left arm
- Hand pockets; Straight hem
- Polyester
- Machine wash
- Imported

BERLIN BUZZWORDS 2018 JUNE 10-12

UNBXD

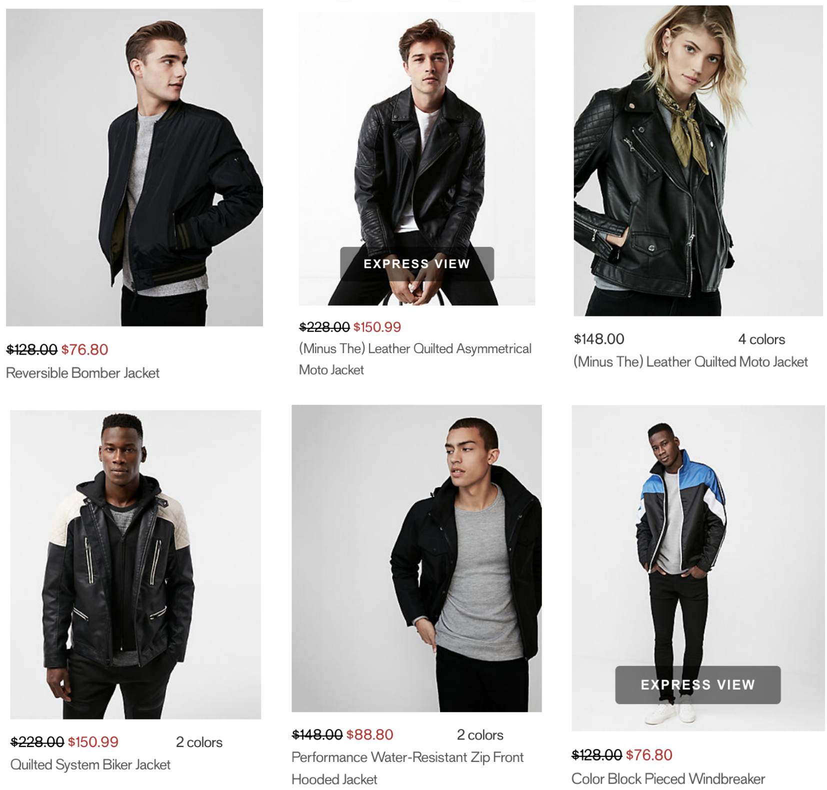
Synonyms improve recall

Thinking about buying Christmas Pajamas ?

Relevant products might be organized under category → “festive wear”, “christmas PJs”, “festive pajamas”

- Synonymy variants that are also contextual to query most often more useful in ecommerce
  - **Conventional synonyms:** pullover ⇔ sweater
  - **Strongly Related words**: printers ⇔ laserjet
  - **Spelling/lemma variants**:
    - wireless enabled phone ⇔ phone with wifi,
    - “packers tee” ⇔ “green bay packers t-shirt”
- Boolean queries won't find Christmas pajamas in ad-hoc categories and this is often the case
  - e.g. “festive wear”, “christmas PJs”, “festive pajamas”

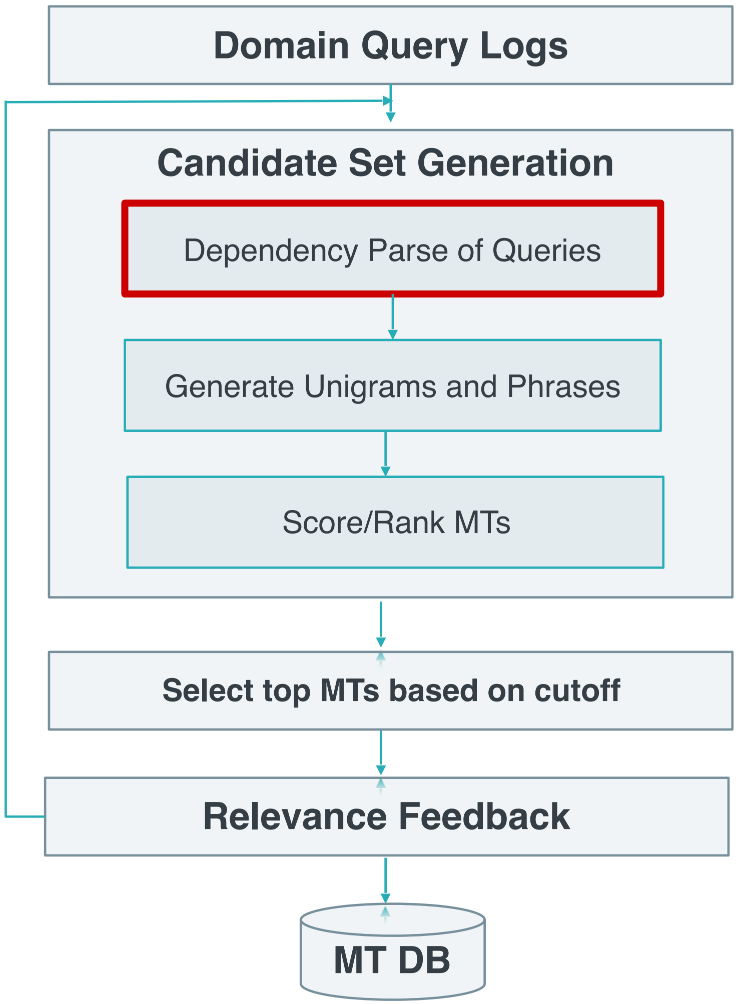
Query Understanding



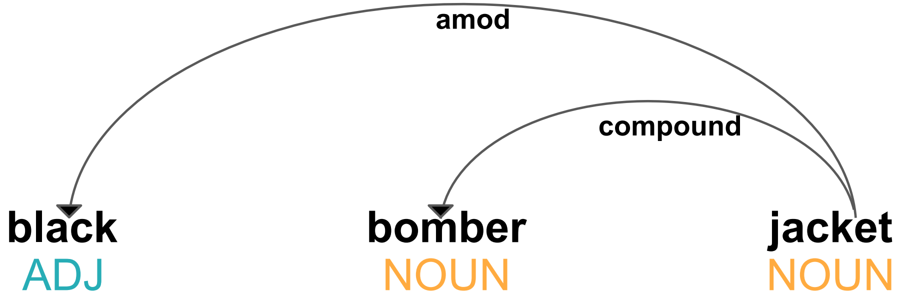
- Query = black bomber jacket
- MT recognizer(black bomber jacket) → (bomber jacket)
- Synonym augmenter(bomber jacket) → (Moto Jacket, Motorcycle Jacket, Biker Jacket, windbreaker, Hooded Jacket)

UNBXD

MT Generation steps

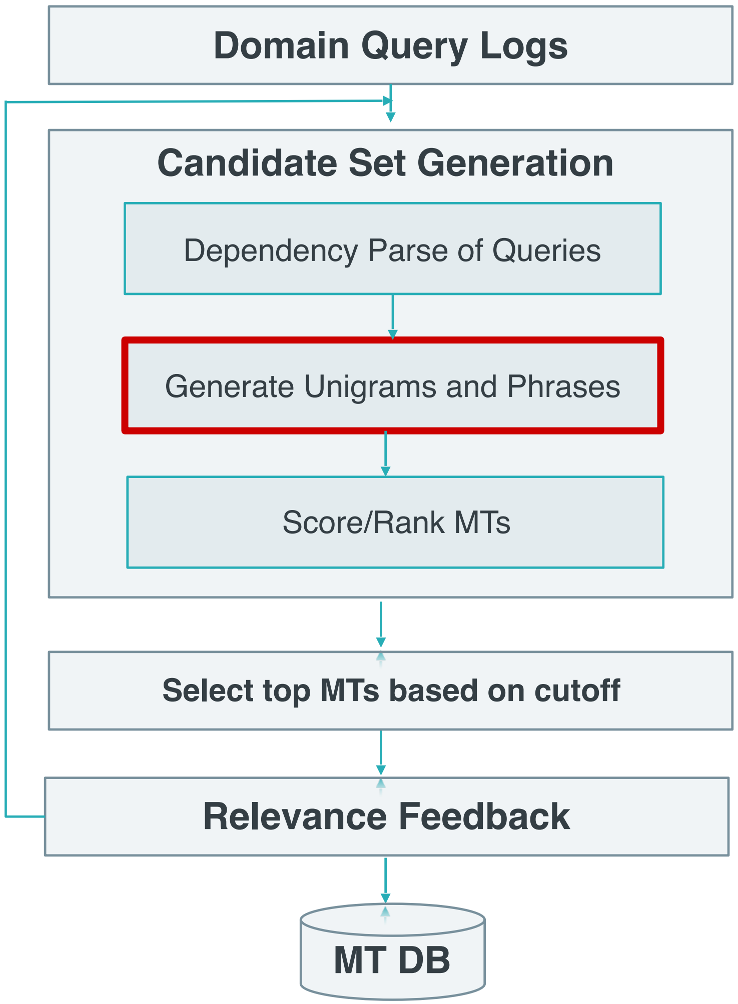


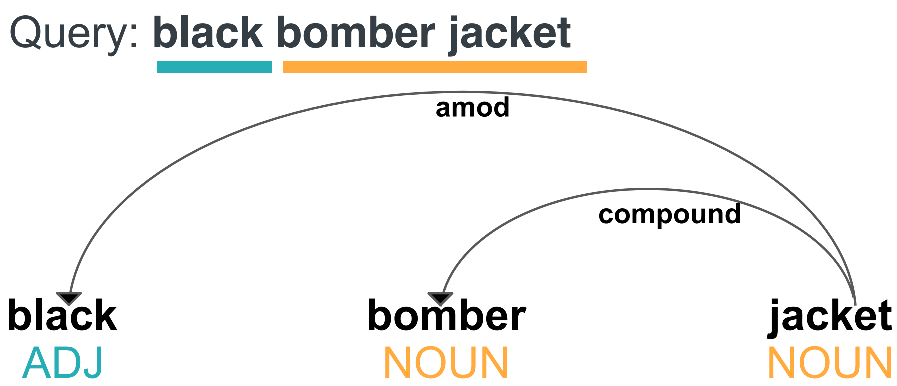
Query: **black bomber jacket**



- amod ⇔ adj-noun-modifier-relation : adj that modifies the meaning of the noun
- compound ⇔ noun-compound-noun relation: noun that modifies the meaning of another noun
- More than **85%** top queries: amod
- More than **53%** top queries: compound

MT Generation steps

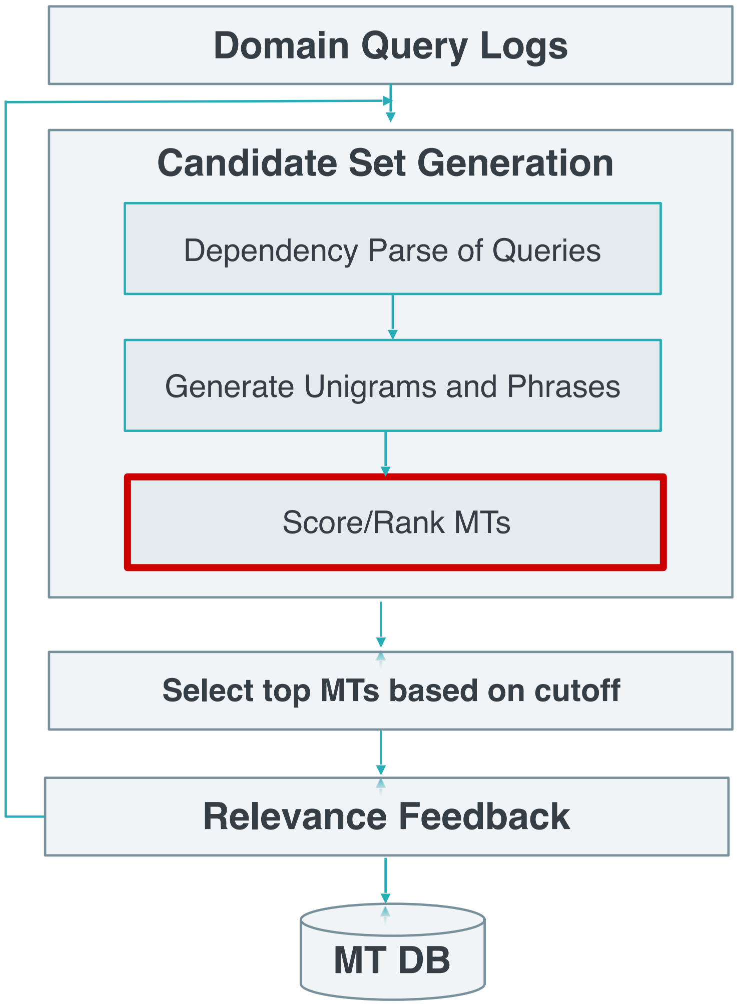




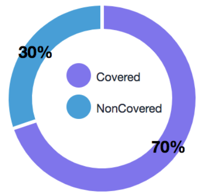
Generate MTs using the Dependency Parser

- Unigram MTs ⇔ root of ‘amod’ relationships e.g jacket in ‘black-amod-jacket’
- Phrase MTs ⇔ nouns connected with compound e.g ‘bomber jacket’ in ‘bomber-compound-jacket’

MT Generation steps

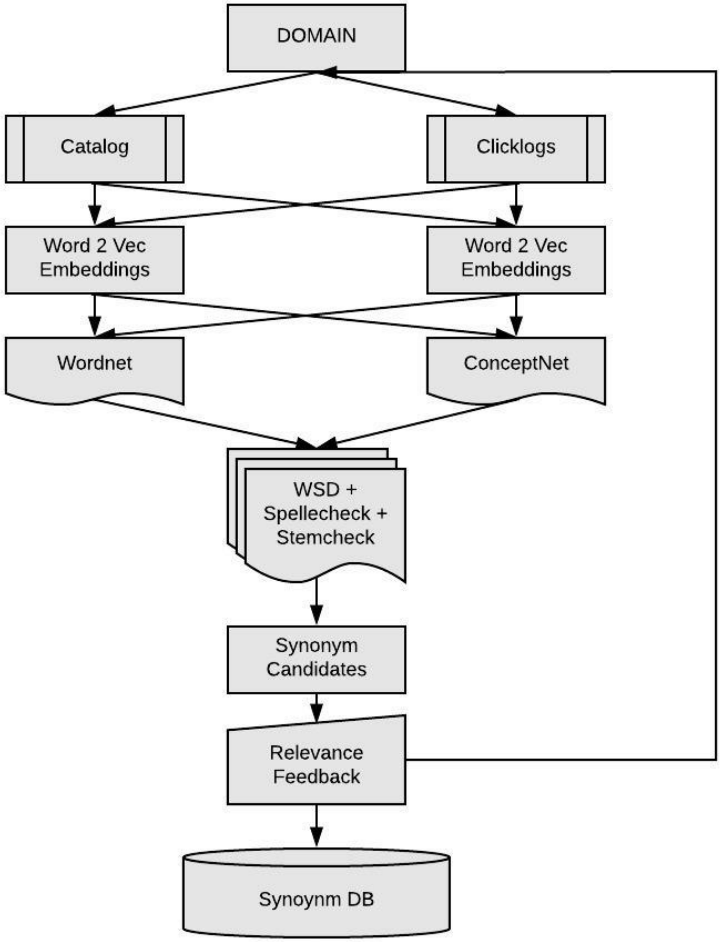


- Score/rank generated MT based on actual_query_coverage + count(root_of_amodl is_a_compound)



- Grammatically incorrect queries `jacket bomber black` will generate `black` as MT but low count(root_of_amodlis_a_compound), hence rejected

Synonym Generation Pipeline



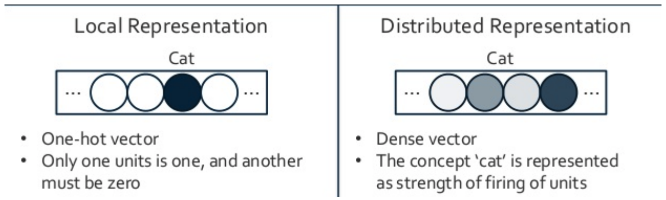
1. Build **Local Corpus** per domain OR per customer
    a. Local corpus ⇔ catalog + sample queries
2. Train **word vector embedding** of local corpus
3. Generate MTs from local corpus to be used as keys
4. Generate synonyms
    a. Input MT list items to a **Global Corpus**(WordNet/ ConceptNet)
    b. Input MT list items to Local Word2Vec.
5. Pipe synonyms word sense disambiguator (**WSD**) in embedding space:
    a. Basis ⇔ Distance(synonymSubspace, querySubspace)

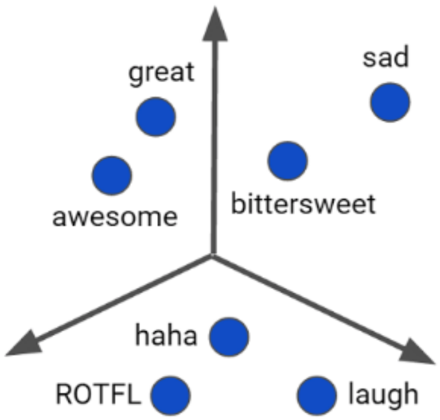
       distance
6. Reject winning candidates based on misspellings and stemmed duplicates

Language, meaning , context, machine

- Humans understand language very well, machines do not.
- Given millions of document being generated per day impossible for a human to categorize, classify or translate all of them. Hence we need to convert them to a format that helps machine do NLP.
- Representation of words which captures context of use, lexical ambiguity, semantic relationships is called Word Vector Embedding and it represents each word in the Vocabulary as a n-dimensional vector of floats that a machine understands.

What are Word Vector Embeddings?





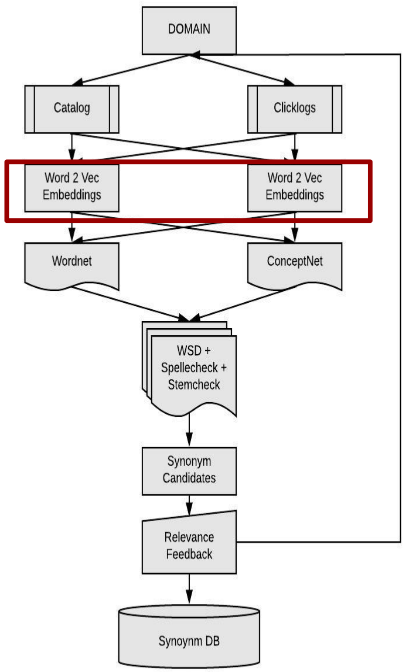
Word Embedding Vectors (dense, continuous space)

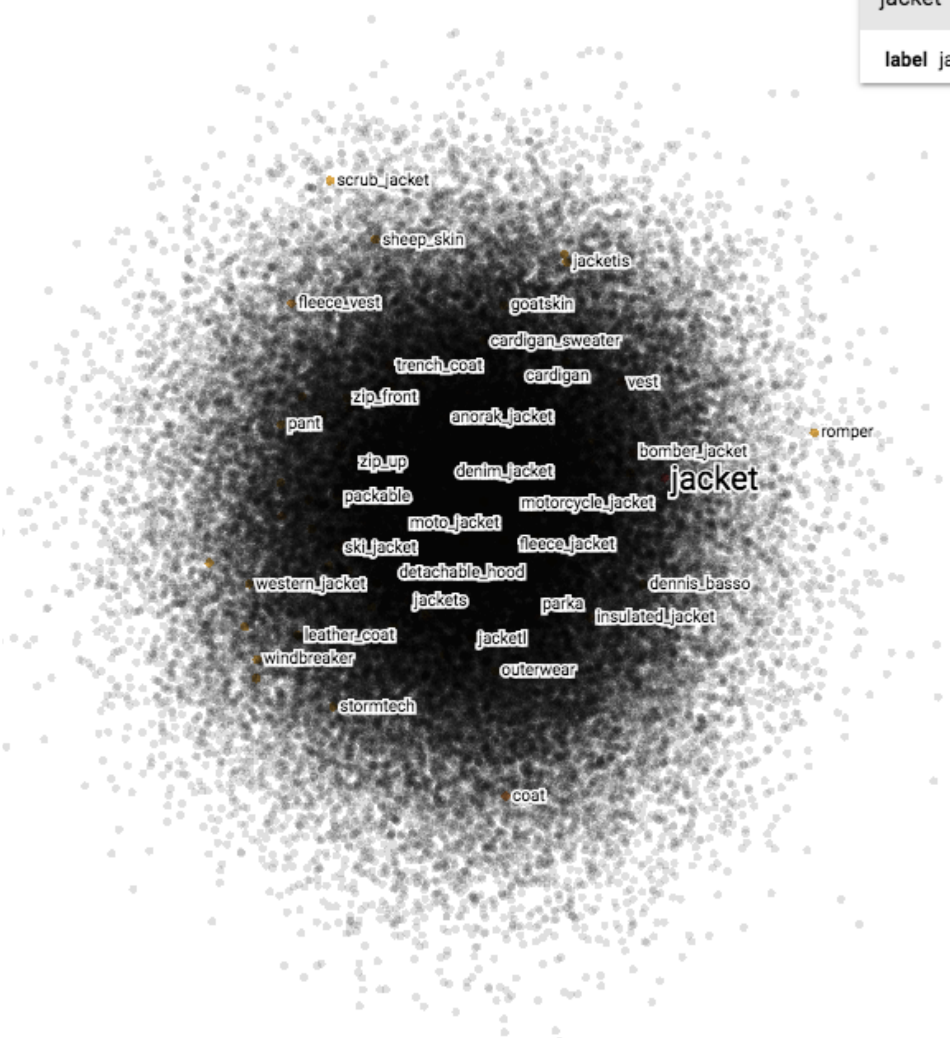
- Words as symbols carry little information
- Hinton : distributed representation
  - Represent word as word= f(contextual words)
- Word vector embedding
  - Word vector = f(contextual words) in optimal dimensions
  - Captures context /lexical ambiguity/semantics difficult to
  - model otherwise
- 2 neural network learned models
  - CBOW(given context → predict missing word)
  - Skipgram (given word → predict context)
  - We have used **Google Word2Vec**
  - **(CBOW + Skip gram) Neural Net Embeddings**

BERLIN BUZZWORDS 2018 JUNE 10-12

UNBXD

PCA Of Fashion Word Vector Embeddings





jacket

label jacket

Search: jacke .* by label

neighbors 100

distance COSINE EUCLIDEAN

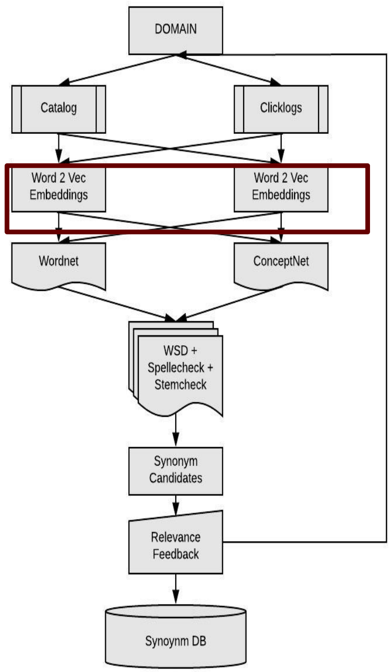
Nearest points in the original space:

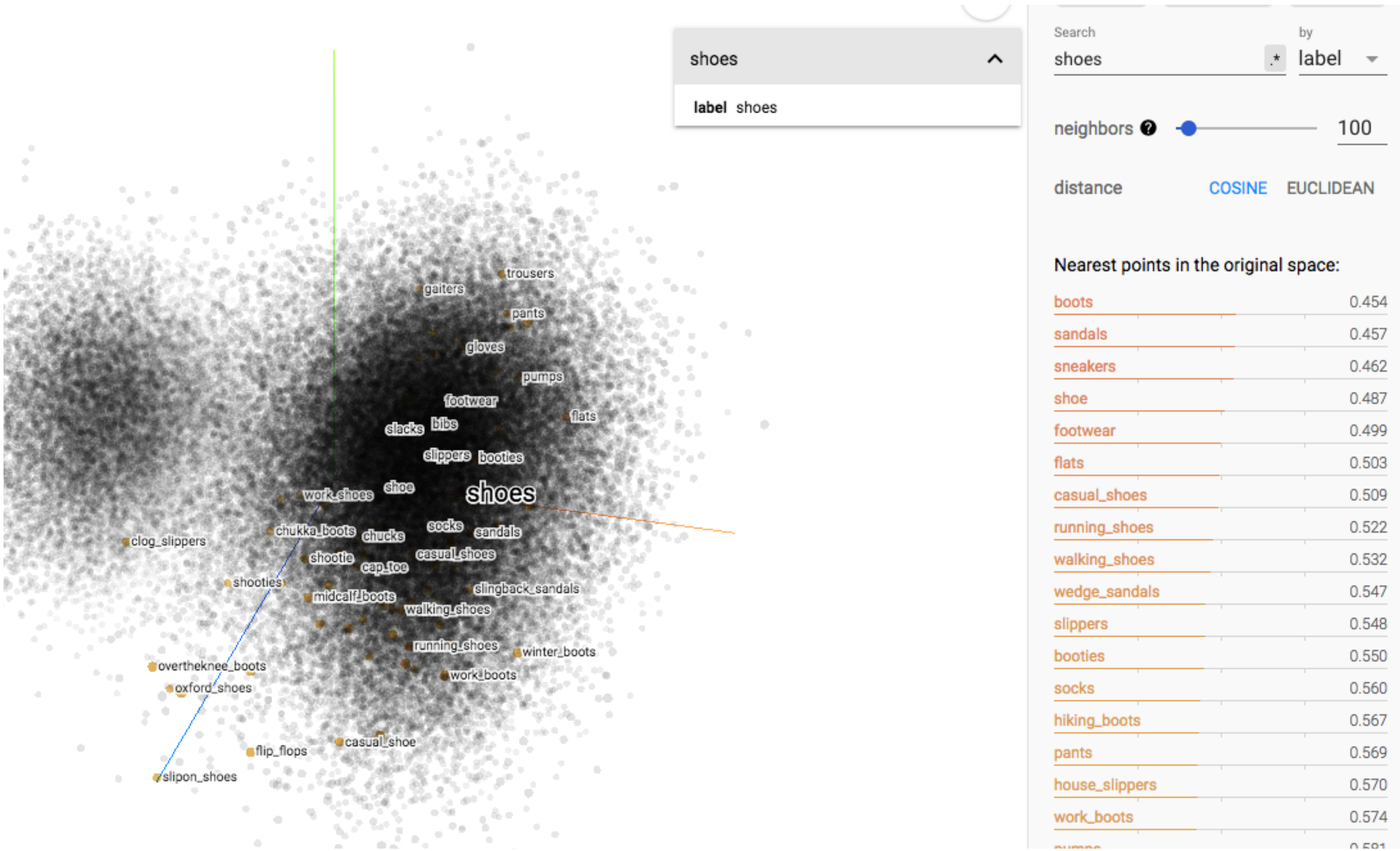
vest	0.447
fleece_jacket	0.463
leather_jacket	0.469
bomber_jacket	0.470
moto_jacket	0.474
parka	0.481
trench_coat	0.485
hooded_jacket	0.489
padded_jacket	0.496
puffer_jacket	0.506
coat	0.520
puffer_coat	0.526
motorcycle_jacket	0.532
puffer_vest	0.540
anorak_jacket	0.543
blazer	0.545
biker_jacket	0.546
windbreaker_jacket	0.547

BERLIN BUZZWORDS 2018 JUNE 10-12

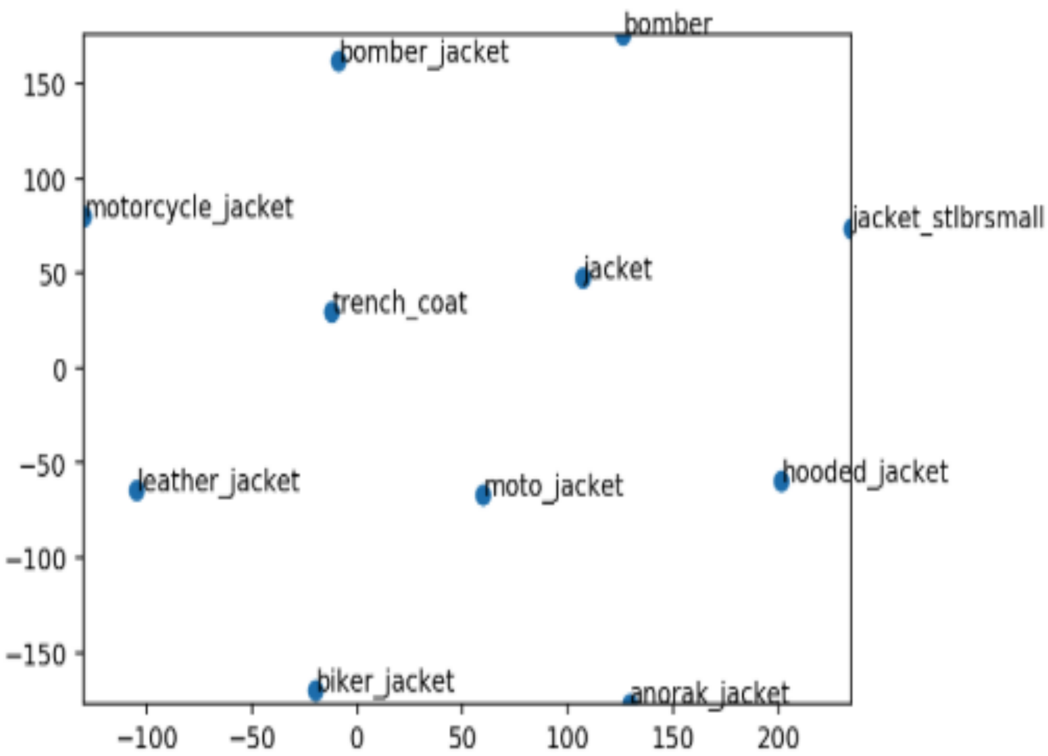
UNBXD

PCA Of Fashion Word Vector Embeddings

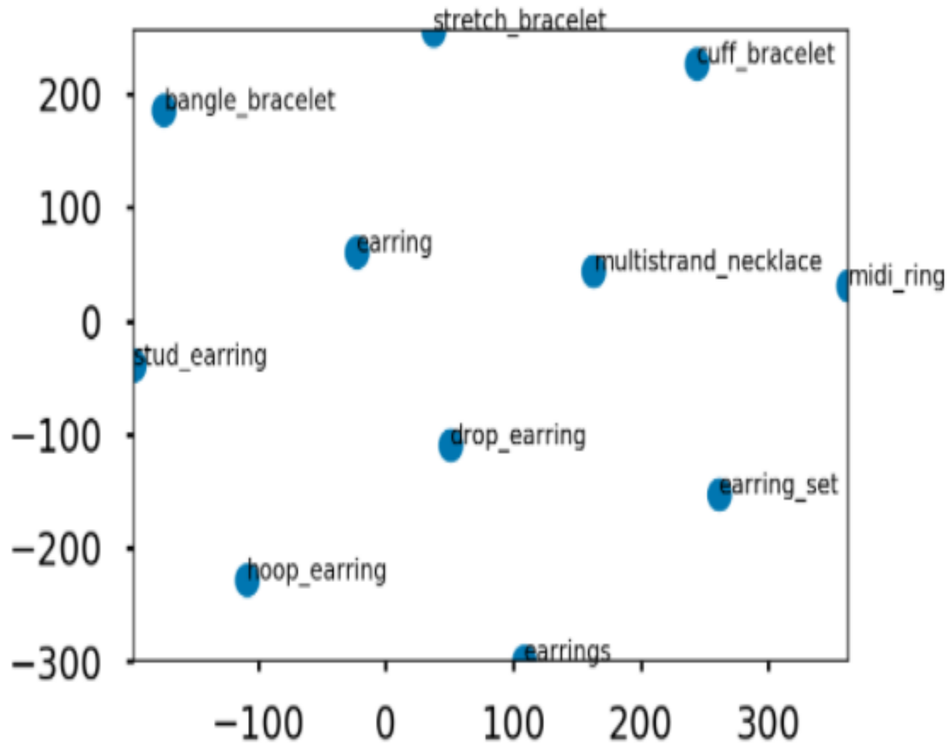




Word2Vec synonymy in catalog space

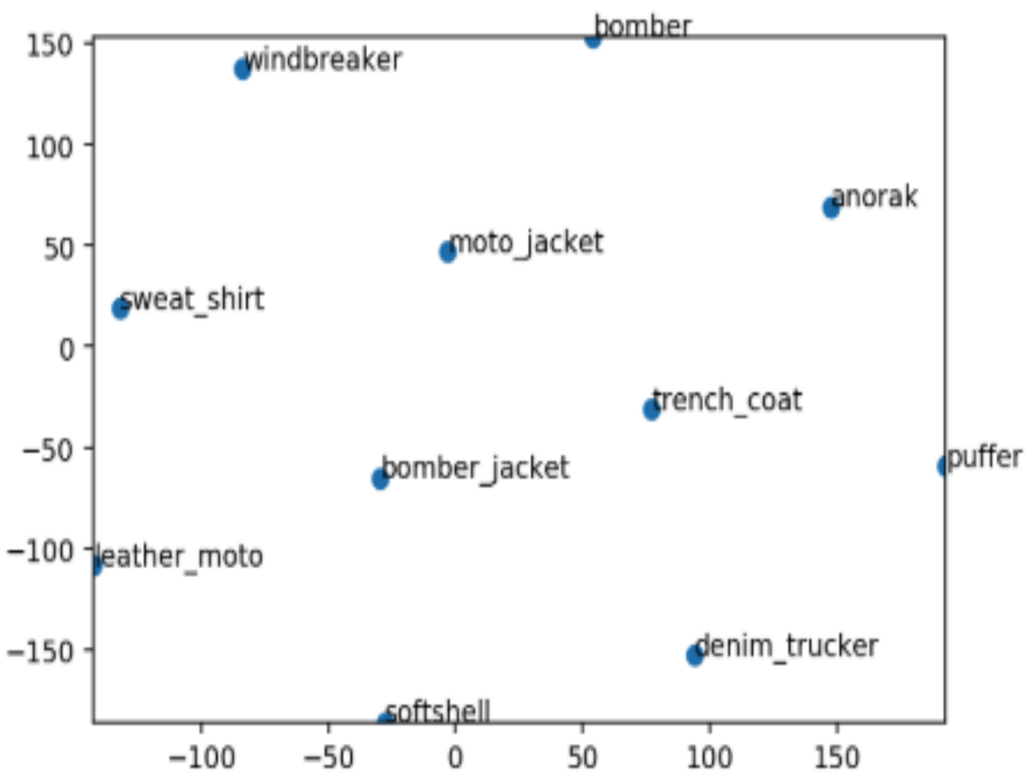


$w2vec_c$ ("bomber jacket")

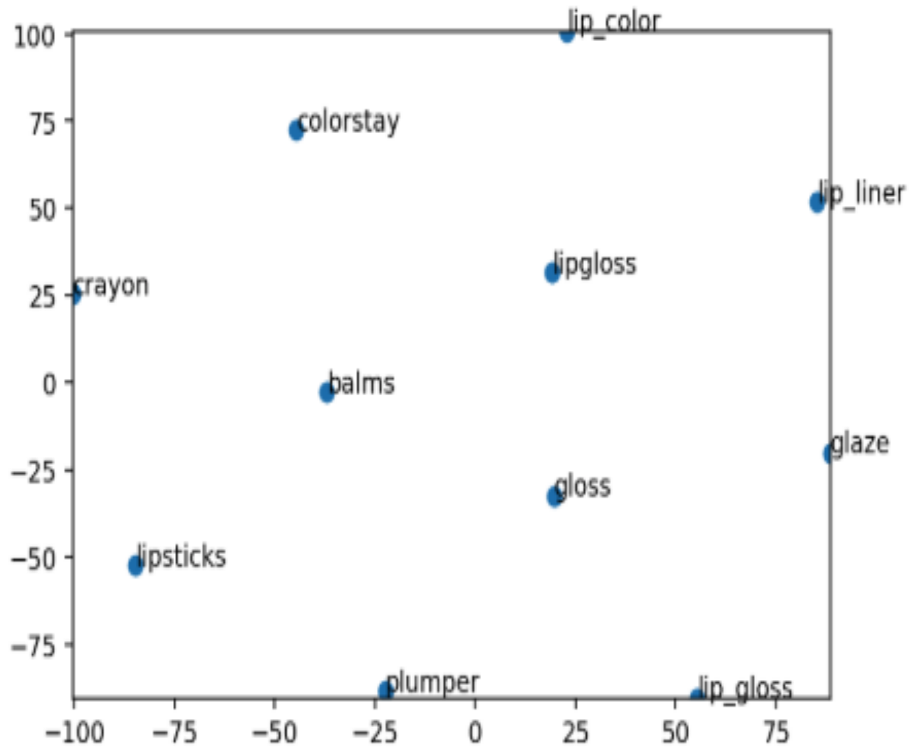


$w2vec_c$ ("earrings")

Word2Vec synonymy in user query space

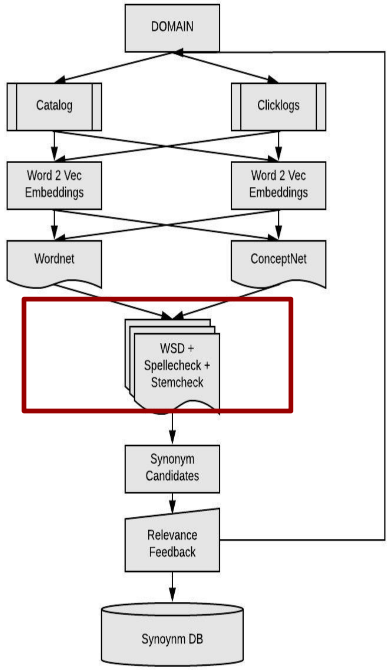


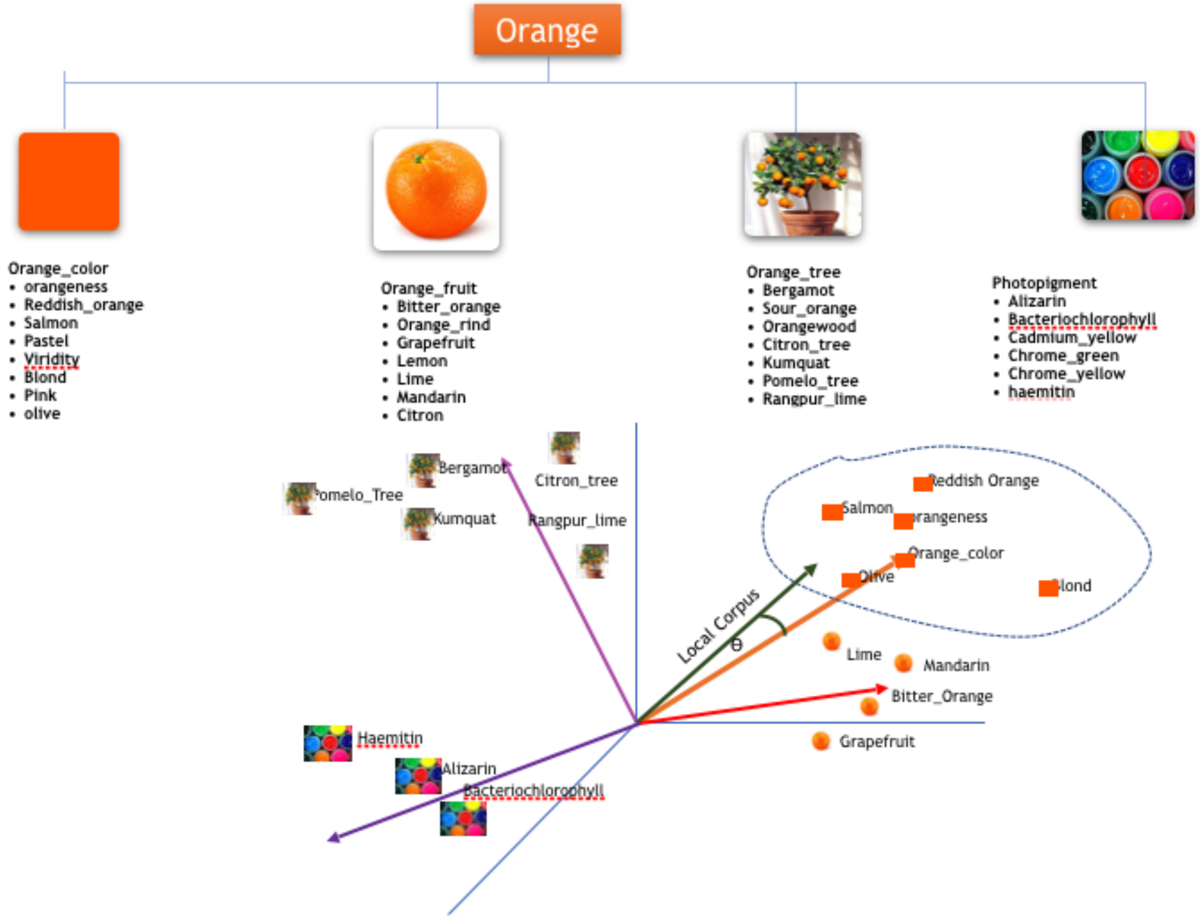
w2vec$_u$ ("bomber jacket")



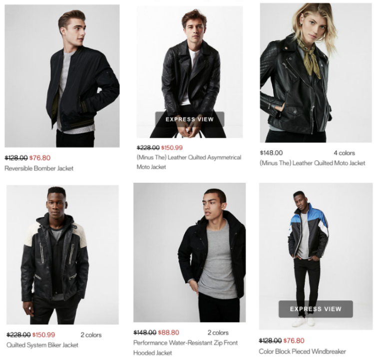
w2vec$_u$ ("gloss")

Performing WSD using Word2Vec





Semantic Retrieval Summary



- Query = black bomber jacket
- MT recognizer(black bomber jacket) → (bomber jacket)
- Synonym augmenter(bomber jacket) → (Moto Jacket, Motorcycle Jacket, Biker Jacket, windbreaker, Hooded Jacket)

- Query → Dependency Parsing + Scoring → MT
- Word2vec on local corpus
- MT as key → Word2vec catalog synonym + clicklog synonym
- MT as key → conceptnet/wordnet synonym candidates + WSD
- MT OR SYNONYM → Final Query
- Final Query → Edis Max Solr Query

Conclusions and Future Work

- We intend to train our own dependency parser using Deep Learning for further boosting MT recognition algorithm
- We intend to extend MT-SYNONYM learning from one client to other clients and finally over one domain
- We intend to improve and simplify the vector algebra operations on synonymy vector
- We intend to further tune and improve performance figures using mapreduce based Word2Vec training
- Implement relevance feedback to autocorrect good synonym and MT pairs vs noisy pairs

Thank you!

Team members : Gururaj Desai, Soumik Chatterjee, Prasad Joshi

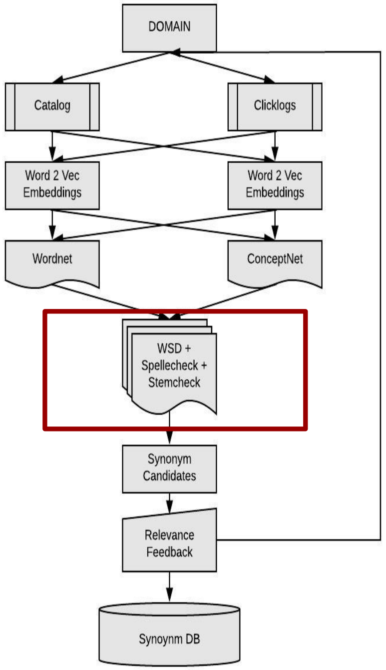
Twitter : @ArpanmGupta, @seinjuti

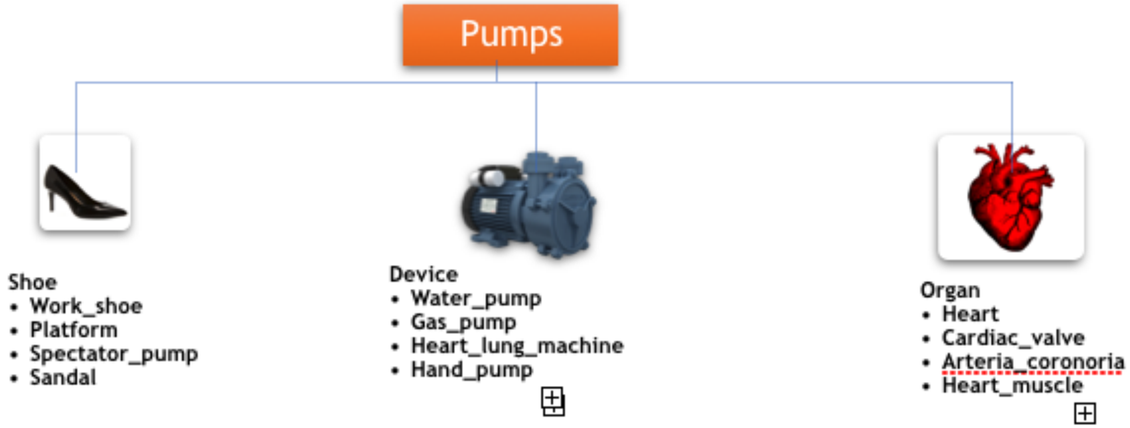
Email : arpan@unbxd.com, seinjuti.chatterjee@unbxd.com

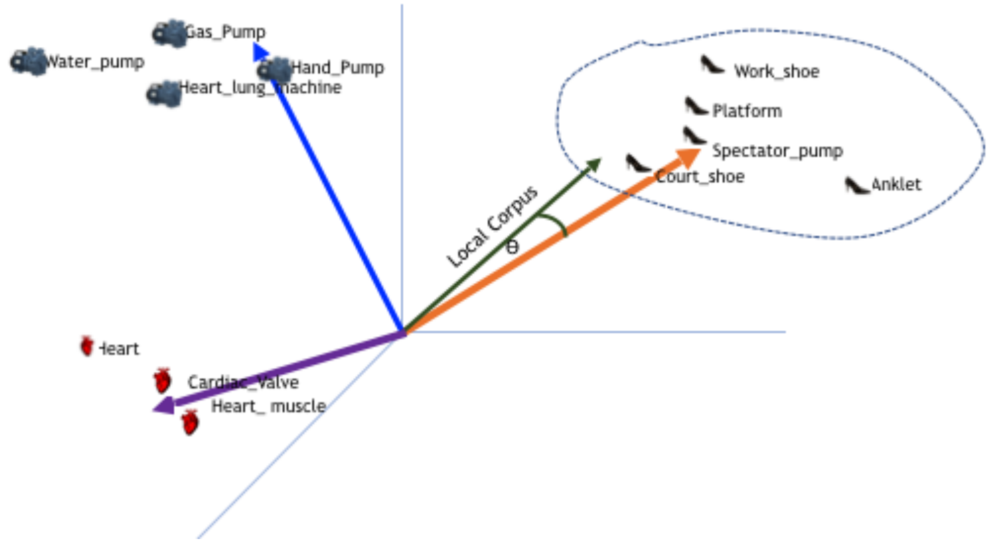
Questions ?

Addendum

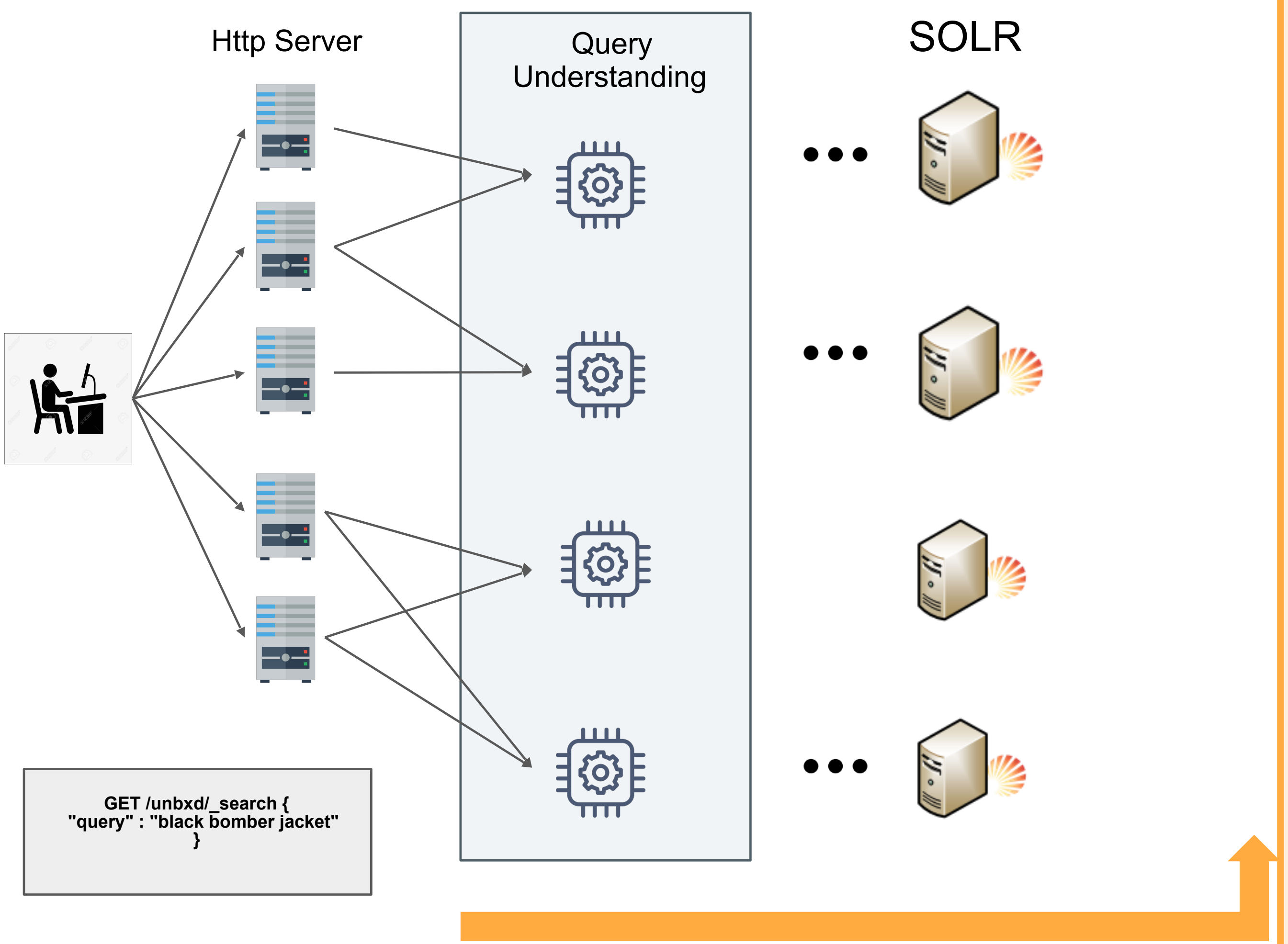
Performing WSD using Word2Vec - Ex2







Search Stack (Query Understanding Layer)




```
GET /unbxd/_search {
"query" : "black bomber jacket"
}
```

MT Builder

```
GET /unbxd/_search {
  "query" : "black bomber jacket",
  "mt" : "bomber jacket"
}
```

Synonym Builder

```
GET /unbxd/_search {
  "query" : "black bomber jacket",
  "mt" : "bomber jacket",
  "synonym" : "motorcycle
jacket, windbreaker jacket
}
```

Asterix Service

```
GET /unbxd/_search {
  "query" : "black bomber jacket",
  "modified_query" : "black +
( "bomber jacket" | "motorcycle
jacket" | "windbreaker jacket" )"
}
```

Brewer Service

```
GET /unbxd/_search {
  "query" : "black bomber jacket",
  "modified_query" : "black +
( "bomber jacket" | "motorcycle
jacket" | "windbreaker jacket" )",
"MM" : 0,
"boost":"seersucker_brand"
}
```

Custom Solr Plugin

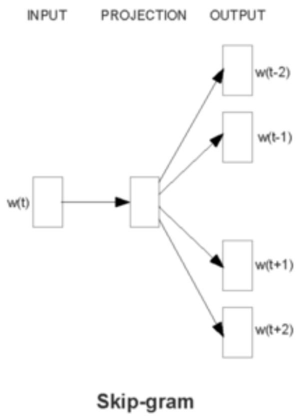
```
GET /unbxd/_search {
  "query" : "black bomber jacket",
"edismax_query":DisjunctionMa
xQuery(title: black)
+DisjunctionMaxQuery( (title:
"bomber jacket"
title:"motorcycle jacket"
title:"windbreaker jacket") )
}
```

QU Layer

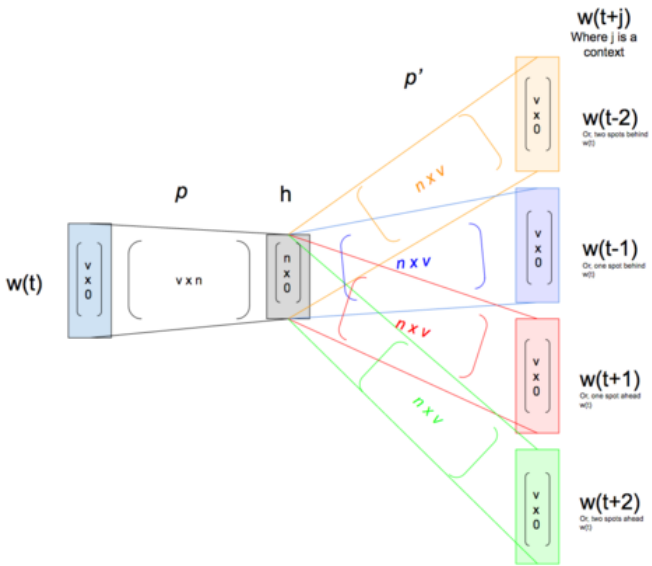
Performance Benchmark of MT and Synonym

On a **8 core 60 GB Linux AWS box network speed@94.5MB/s**

1. Typical mts count per site based on queries ~ 8573 unigrams + bigrams
2. Typical synonyms count per site ~ 6554
3. Typical qps for dependency tree calculation ~ 1000 qps
4. Typical batch qps for conceptnet api based synonym prediction ~ 100 qps
5. Typical batch qps for wordnet api + wsd based synonym prediction ~ 10 qps
6. qps for training word2vec model (multicore multithreaded but single machine) ~ 865K qps
7. Typical accuracy of prediction ~ 10% error rate for known domains like fashion, grocery, home and living, 30% error rate for new domains like autoparts



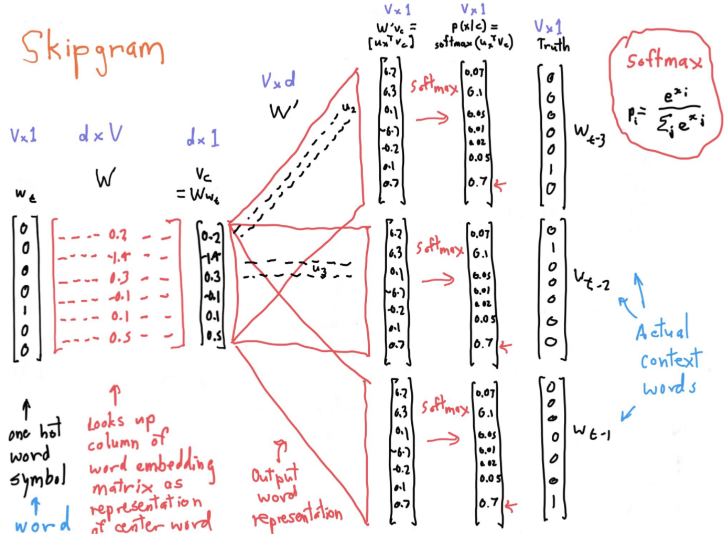
Original diagram from Mikolov et al (2013)



Extended diagram identifying matrix dimensions

Google Word2Vec (CBOW + Skipgram)

The Skip-Gram Algorithm:



Objective function:

Maximize the probability of any context word given the current center word:

$$J'(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} p(w_{t+j} \mid w_t ; \theta)$$

Negative Log Likelihood

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log p(w_{t+j} \mid w_t)$$

$$p(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$