Generating Sentences by Editing Prototypes

Kelvin Guu*, Tatsunori Hashimoto*, Yonatan Oren, Percy Liang

TACL 2018, appeared at ACL 2018

\begin{**Overview**}

Goal: sentence generation

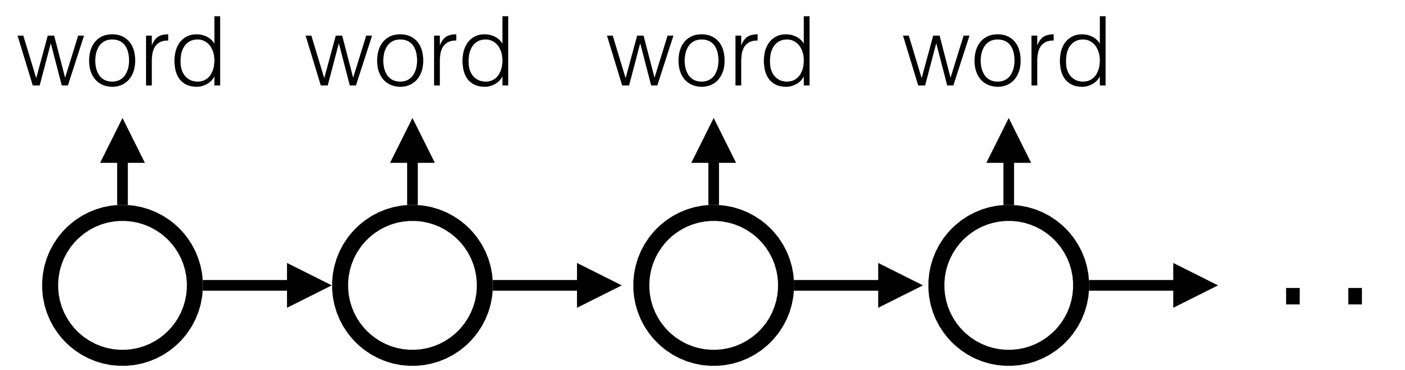
$p(y) \longrightarrow$ y = "stocks fell by 2 percent"

x = "死马当活马医" $\xrightarrow{p(y \mid x)}$ y = "beating a dead horse"

x = "how are you?" $\xrightarrow{p(y \mid x)}$ y = "pretty good, you?"

The status quo

- left to right
- word by word



Train on wide output distributions

- low diversity
  - the generic utterance problem
  - (*"I don't know", "I'm sorry"*) [Li+ 2016, Serban+ 2016, Ott+ 2018]
- no semantic control [Hu+ 2017]

Approach: prototype, then edit

Overpriced , overrated , and tasteless food .
The food here is ok but not worth the price .
I definitely recommend this restaurante .

Sample from the training set

Prototype

The food here is ok but not worth the price .

prototype controls rough semantics

guarantee diversity

Edit using attention

Generation

The food is mediocre and not worth the ridiculous price .

The food is good but not worth the horrible customer service .
The food here is not worth the drama .
The food is not worth the price .

seq2seq editor injects variation

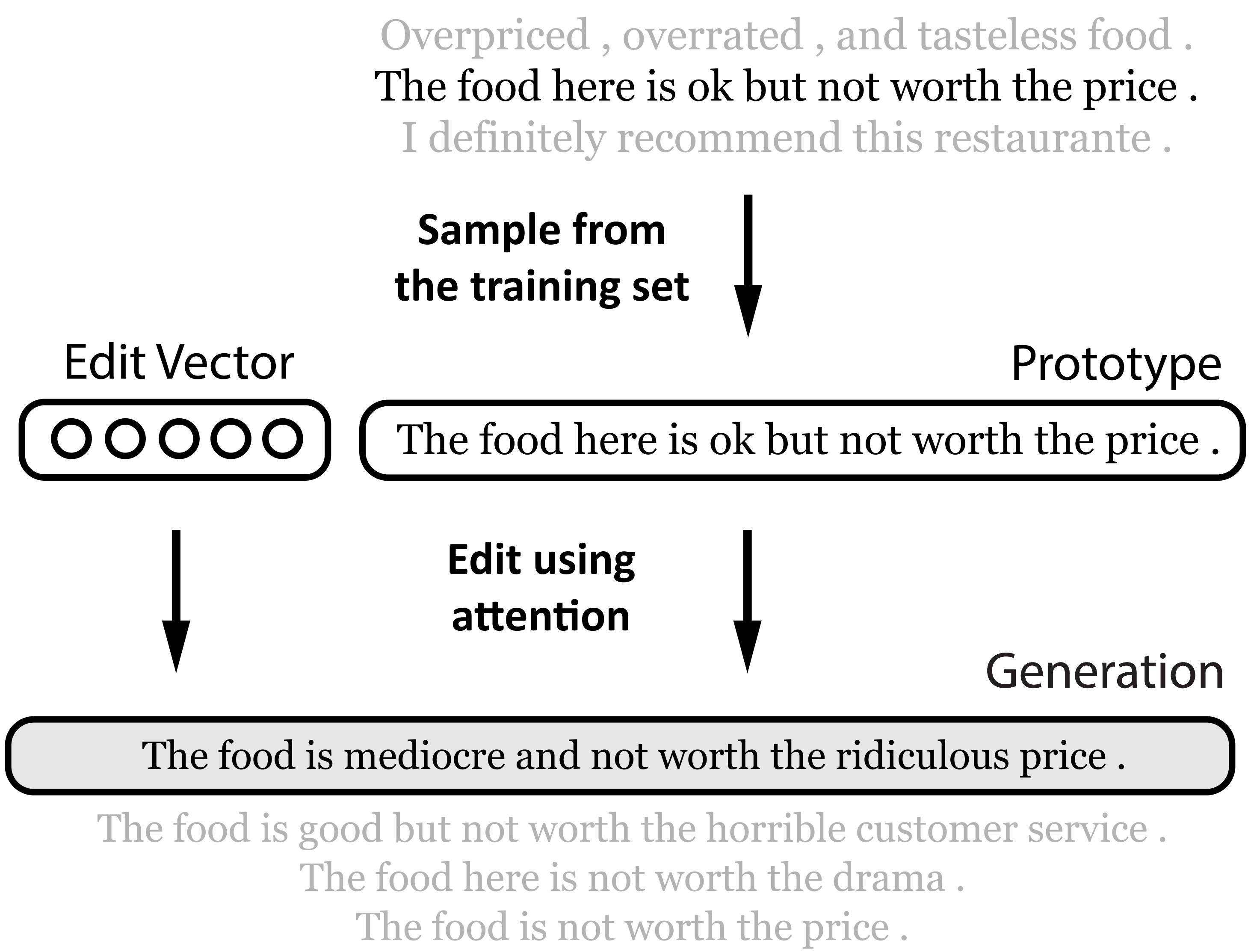
Overview of results

- **More diverse generations**
- **Higher quality generations** (Mechanical Turk)
- **Better perplexity** (BillionWord, Yelp reviews)
- **Seq2seq edits are semantically interpretable**
    - preserve semantic similarity
    - can be used to perform sentence-level analogies

\end{**Overview**}

\begin{**Approach**}

prototype, then edit **(formally)**



$$z_p \sim p_{\text{proto}}$$

$$z_e \sim p_{\text{edit}}$$

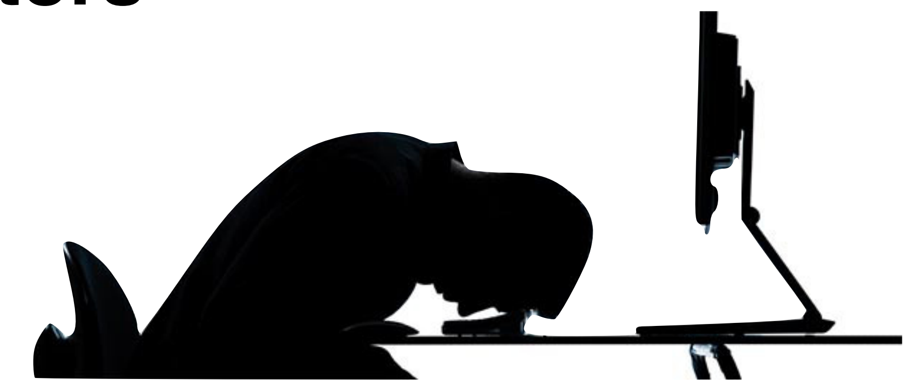
$$y \sim p_{\text{editor}}\left(y \mid z_p, z_e\right)$$

y = output sentence **z**$_p$ = prototype sentence **z**$_e$ = edit vector

Intuitions

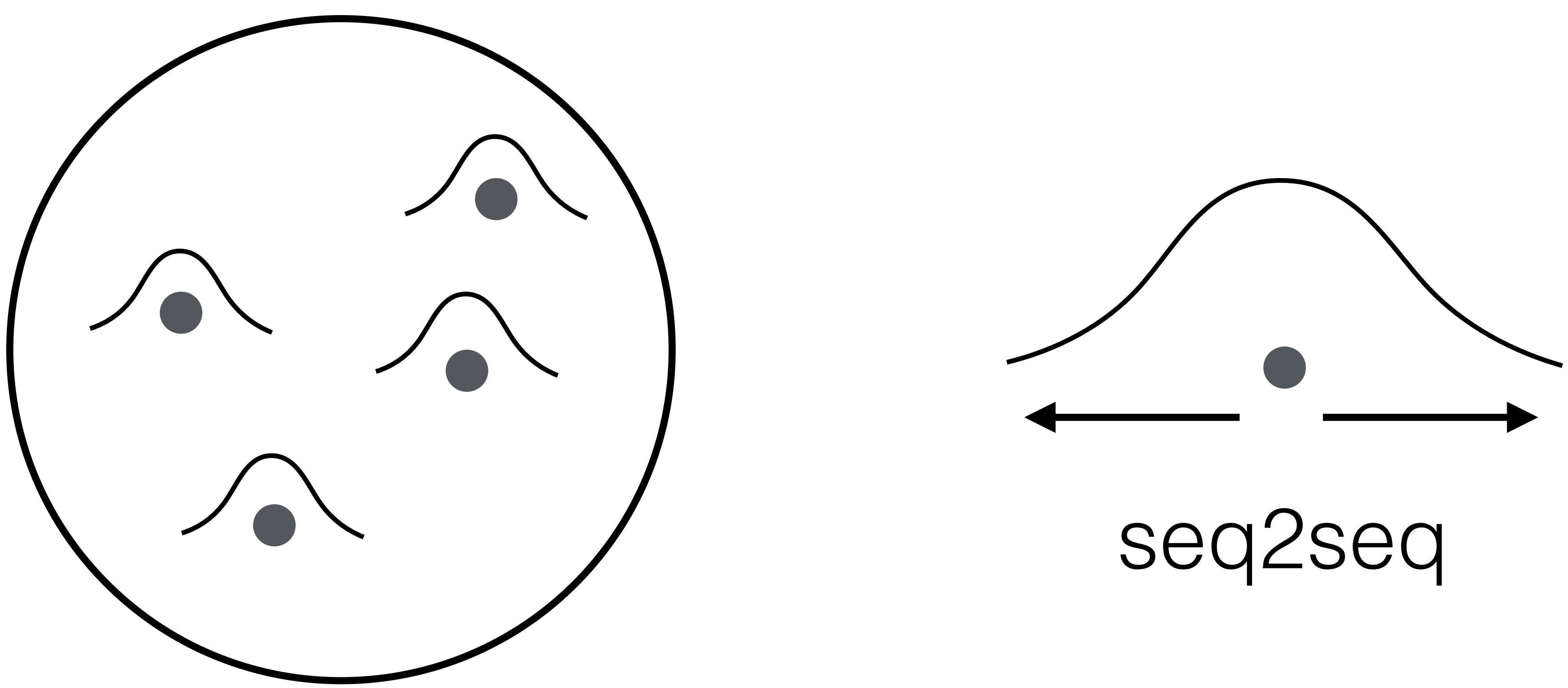
humans are not pure left-to-right generators

- we write a first draft, then edit
- we use templates
- we plagiarize



semi-parametric statistics

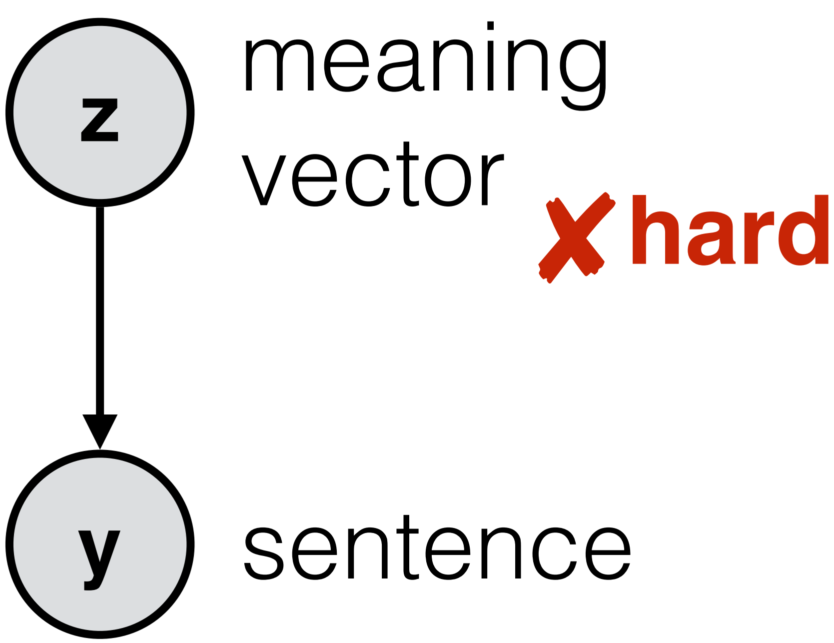
- we are doing **kernel density estimation** over sentence space

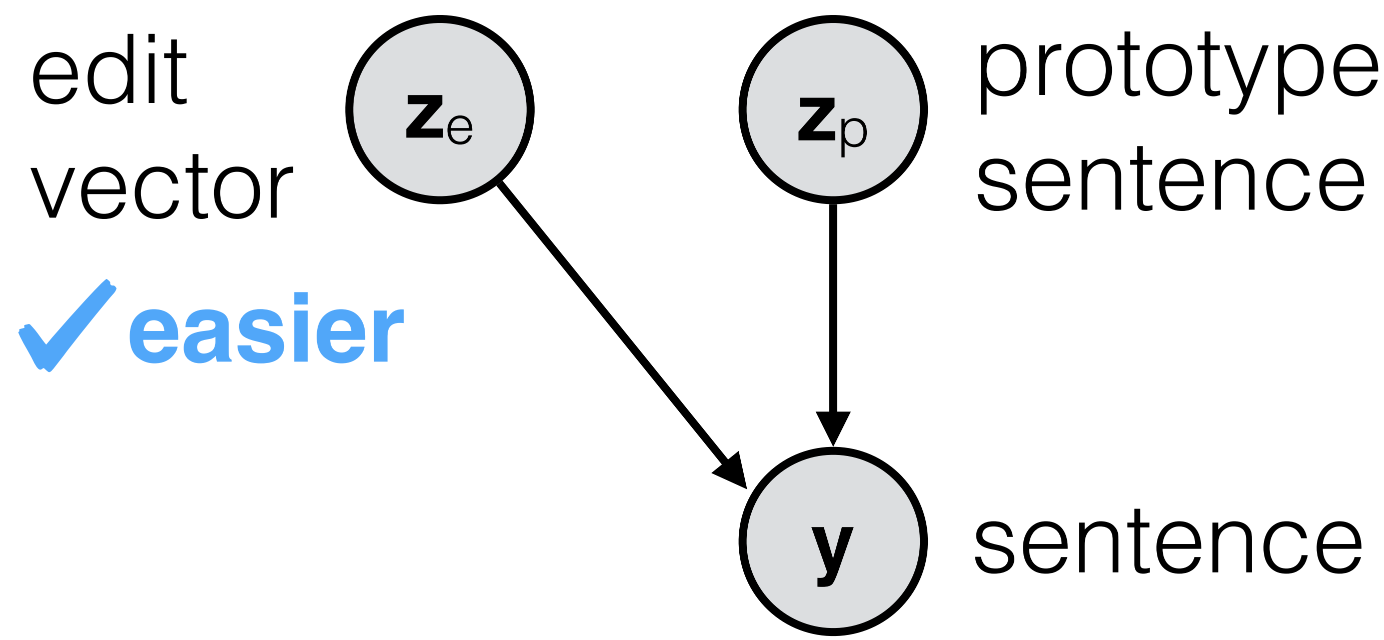


Another intuition

Professor of Computer Science
The University of Texas at Austin

> *You can't cram the meaning of a whole %&!$ing sentence into a single $&!*ing vector!*
>
> [Ray Mooney, ACL 2014]





Training objective

maximize

ELBO

$$p(y) = \sum_{z_p} p(y \mid z_p)\, p_{\text{proto}}(z_p)$$

expensive

ELBO

$$\int_{z_e} p_{\text{editor}}(y \mid z_p, z_e)\, p_{\text{edit}}(z_e)\, dz_e$$

intractable

key tool: **ELBO** (evidence lower bound)

[Dempster+ '77, Jordan+ '99, Kingma+ '13]

- more computationally tractable
- bias towards semantically interpretable edits

y = output sentence **z**$_p$ = prototype sentence **z**$_e$ = edit vector

ELBO (in general)

$$\underbrace{\log p(y)}_{\int_z p(y \mid z)\, p(z)\, dz} \geq \int_z \log p(y \mid z)\, \underbrace{q(z)}_{q(z)}\, dz - KL(\underbrace{q(z)}_{q(z)} \| p(z))$$

q(z)

you choose q(z)

- add helpful biases to the model
- tightness of the lower bound

$$\boxed{q(z) \approx p(z \mid y)}$$

y = output sentence **z**$_p$ = prototype sentence **z**$_e$ = edit vector

Training objective

maximize

ELBO

$$p\left(y\right) = \sum_{z_p} p\left(y \mid z_p\right) p_{\text{proto}}\left(z_p\right)$$

expensive

ELBO

$$\int_{z_e} p_{\text{editor}}\left(y \mid z_p, z_e\right) p_{\text{edit}}\left(z_e\right) dz_e$$

intractable

y = output sentence **z**p = prototype sentence **z**e = edit vector

ELBO on prototypes

$$p(y) = \sum_{z_p} p(y \mid z_p)\, p_{\text{proto}}(z_p)$$

$$\geq \quad \sum_{z_p} \log p(y \mid z_p)\, q(z_p) - KL\left(q(z_p) \,\|\, p_{\text{proto}}(z_p)\right)$$

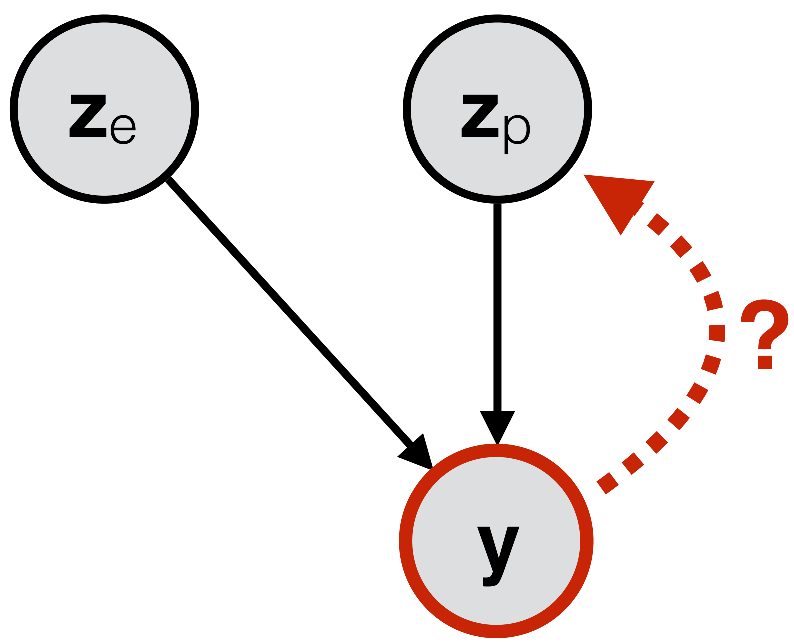
$$q(z_p) \approx p(z_p \mid y)$$ **?**

y = output sentence **z**p = prototype sentence **z**e = edit vector

q(z) over prototypes

Question

$$q\left(z_p\right) \approx p\left(z_p \mid y\right)$$

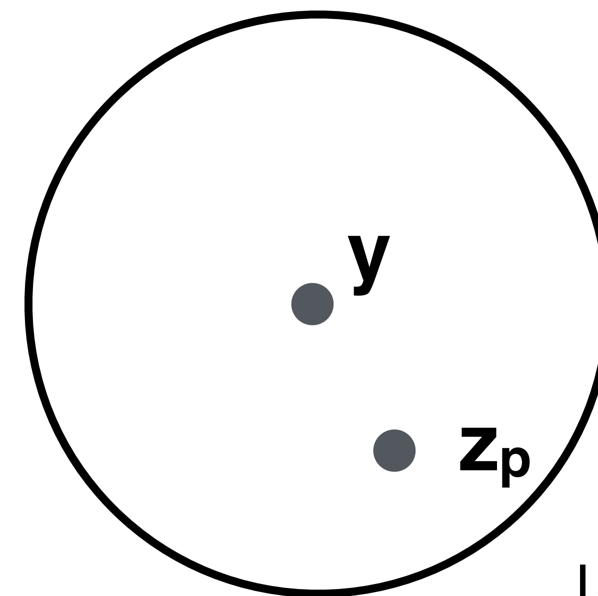


Answer

prototype $\mathbf{z}_p$ was probably not too different from **y**.

$$q(z_p) := \text{Uniform}(\mathcal{N}(y))$$

$$\mathcal{N}\left(y\right) =$$



Jaccard

N(y) = all sentences with high token overlap

y = output sentence $\mathbf{z}_p$ = prototype sentence $\mathbf{z}_e$ = edit vector

q(z) over prototypes

$$\text{ELBO} = \sum_{z_p} \log p\left(y \mid z_p\right) q\left(z_p\right) - KL\left(q\left(z_p\right) \| p_{\text{proto}}\left(z_p\right)\right)$$

$$=$$

$$\frac{1}{\left|\mathcal{N}\left(y\right)\right|} \sum_{z_p \in \mathcal{N}\left(y\right)} \log p\left(y \mid z_p\right) + C$$

Looks like typical **sequence-to-sequence** objective

prototype $\mathbf{z}_p$ —> output $\mathbf{y}$

✓ **bias towards small edits**

✓ **computationally tractable**

$\mathbf{y}$ = output sentence $\mathbf{z}_p$ = prototype sentence $\mathbf{z}_e$ = edit vector

Training objective

maximize

ELBO

$$p\left(y\right)=\sum_{z_p}p\left(y \mid z_p\right)p_{\text{proto}}\left(z_p\right)$$

expensive

ELBO

$$\int_{z_e}p_{\text{editor}}\left(y \mid z_p, z_e\right)p_{\text{edit}}\left(z_e\right)dz_e$$

intractable

y = output sentence **z**$_p$ = prototype sentence **z**$_e$ = edit vector

ELBO on edit vectors

sample $\mathbf{z}_e$ from $\mathbf{q}(\mathbf{z}_e)$

$$\log p\left(y \mid z_p\right)$$

$\mathbf{z}_p$, $\mathbf{z}_e$ —> $\mathbf{y}$

$$\geq \quad \underbrace{E_{z_e \sim q(z_e)}\left[\log p_{\text{editor}}\left(y \mid z_p, z_e\right)\right]}_{\text{reconstruction\_cost}} - \underbrace{KL\left(q\left(z_e\right) \| p_{\text{edit}}\left(z_e\right)\right)}_{\text{KL\_penalty}}$$

reconstruction_cost

measures how well we can reconstruct **y** from prototype $\mathbf{z}_p$ and edit $\mathbf{z}_e$

KL_penalty

measures difference between **q** and edit prior $\mathbf{p}_{\text{edit}}$

y = output sentence $\mathbf{z}_p$ = prototype sentence $\mathbf{z}_e$ = edit vector

ELBO on edit vectors

$$\log p\left(y \mid z_p\right)$$

$$\geq \quad E_{z_e \sim q(z_e)}\left[\log p_{\text{editor}}\left(y \mid z_p, z_e\right)\right] - KL\left(q\left(z_e\right) \| p_{\text{edit}}\left(z_e\right)\right)$$

$$q\left(z_e\right) \approx p\left(z_e \mid y, z_p\right)$$ **?**

y = output sentence **z**$_{\text{p}}$ = prototype sentence **z**$_{\text{e}}$ = edit vector

q(z) over edits

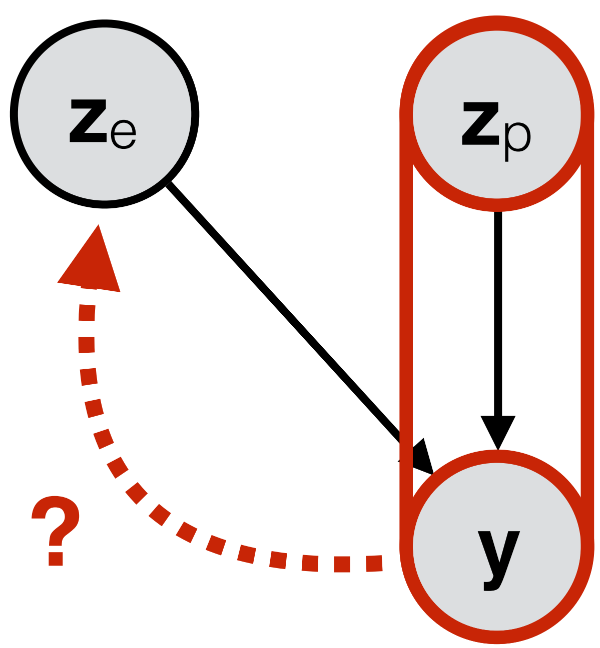
Question

$$q\left(z_e\right) \approx p\left(z_e \mid y, z_p\right)$$

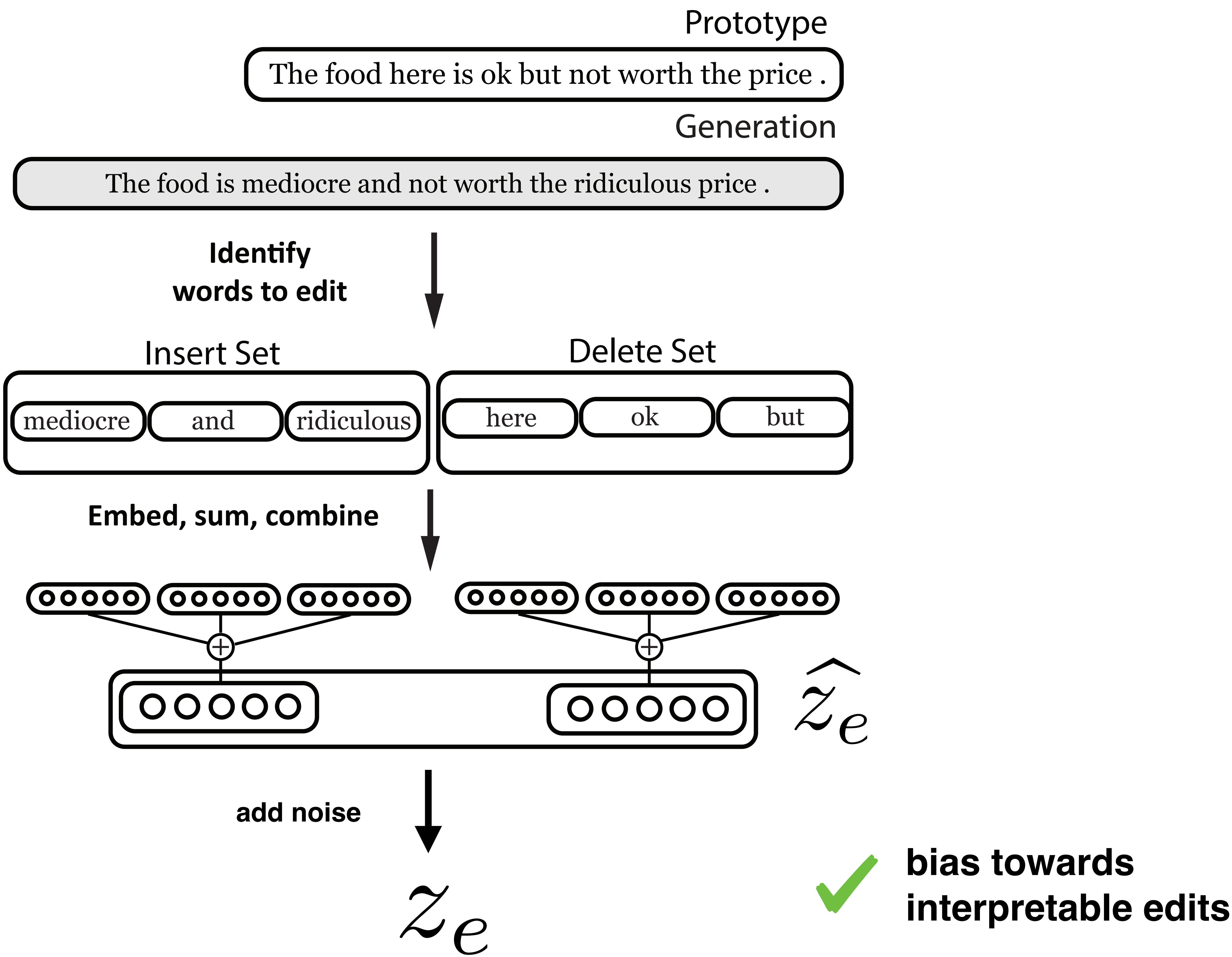
Answer

Compare the two sentences.

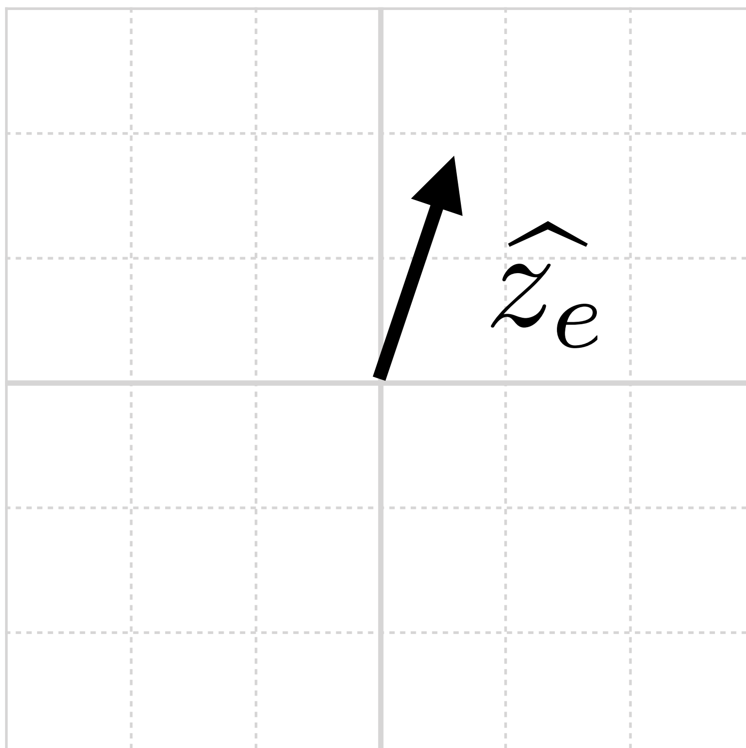
Figure out which words were **inserted** and **deleted**.
Then sum their word vectors.



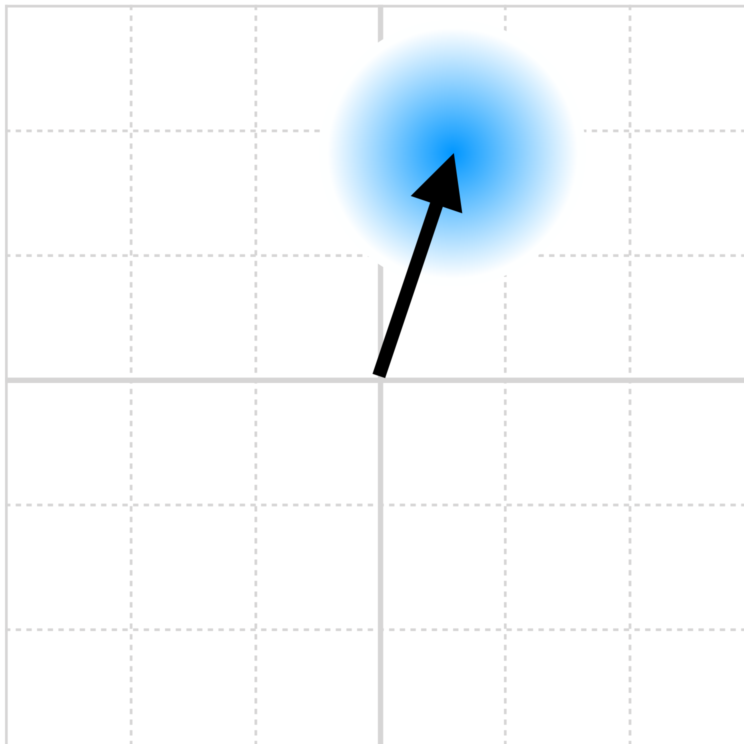
y = output sentence **z**p = prototype sentence **z**e = edit vector


Prototype
The food here is ok but not worth the price .
Generation
The food is mediocre and not worth the ridiculous price .
Identify words to edit
Insert Set
mediocre
and
ridiculous
Delete Set
here
ok
but
Embed, sum, combine
$\hat{z}_e$
add noise
z_e
bias towards interpretable edits

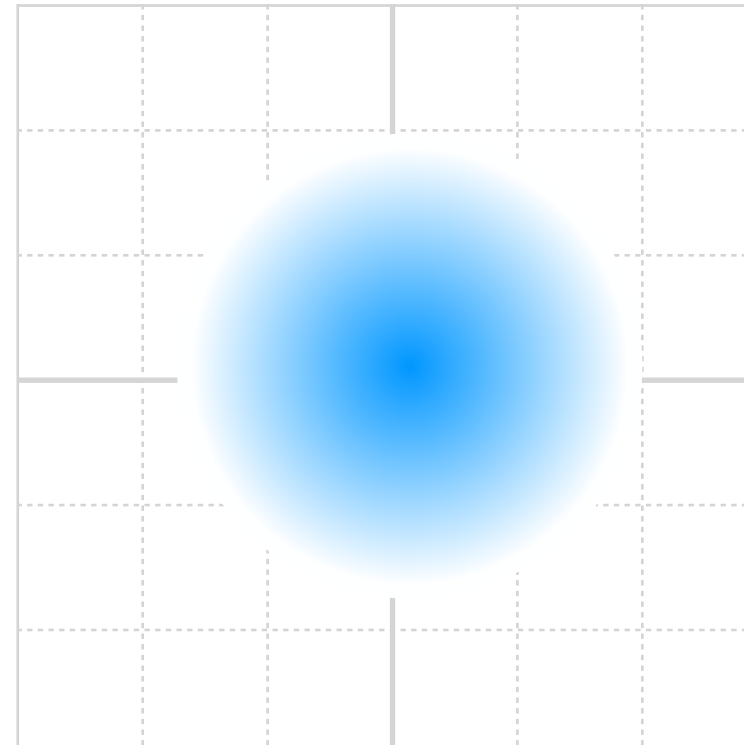
How to add noise to $\widehat{z_e}$?



Standard choice (VAE): Gaussian



$q\left(z_e\right)$



$p_{\mathrm{edit}}\left(z_e\right)$

ELBO = reconstruction_cost - KL_penalty

reparameterization trick (VAEs)

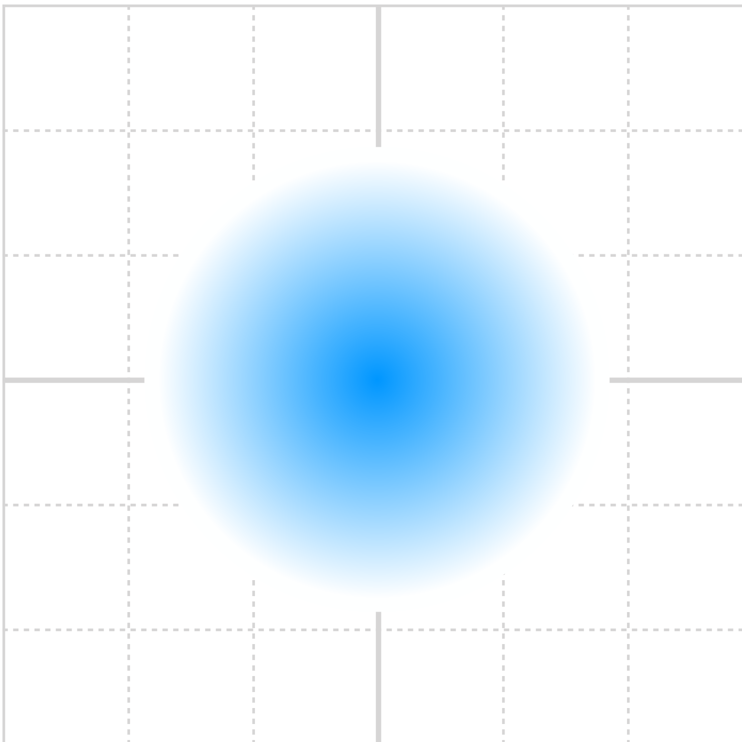
(low-variance MC estimate of gradient)

closed form

computationally tractable

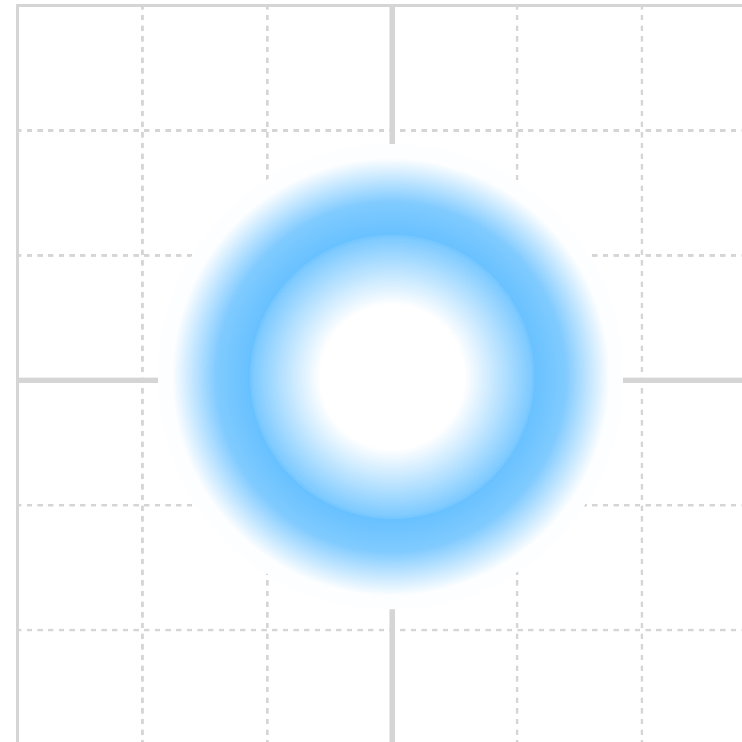
The problem with a Gaussian prior

low-dim Gaussian

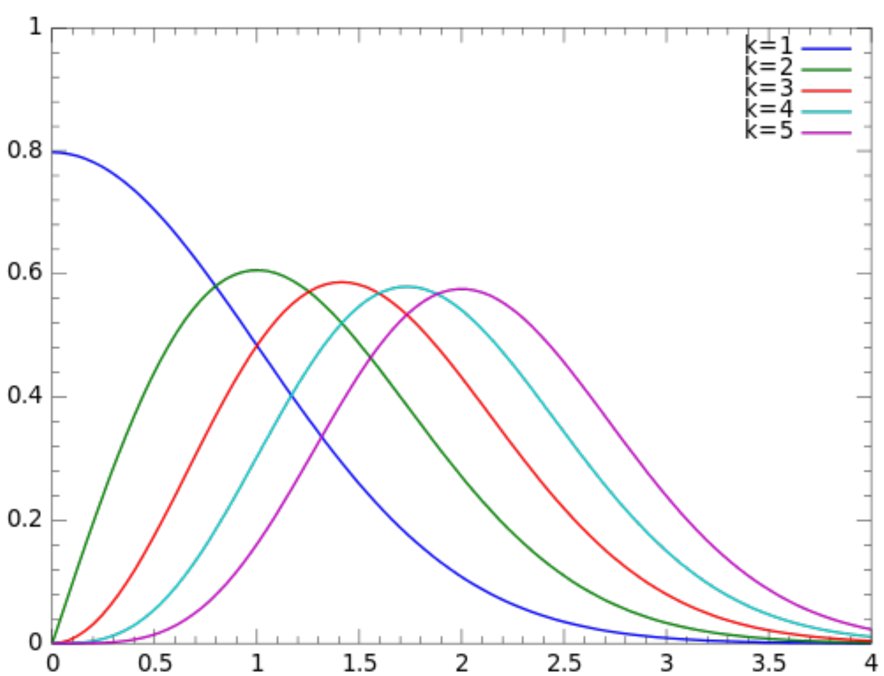


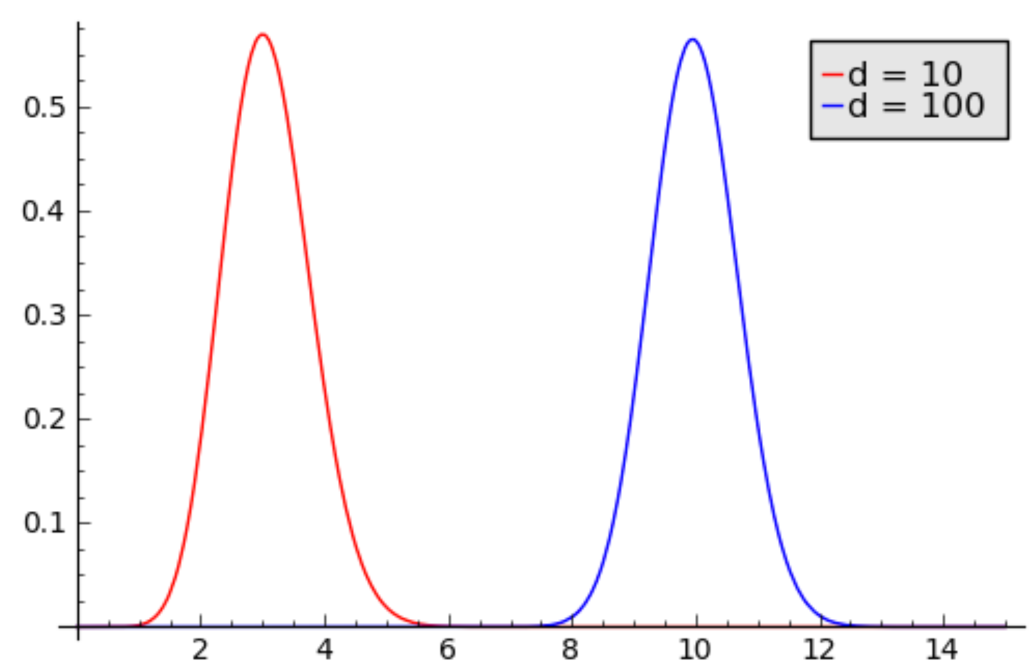
$p_{\text{edit}}(z_e)$

high-dim Gaussian



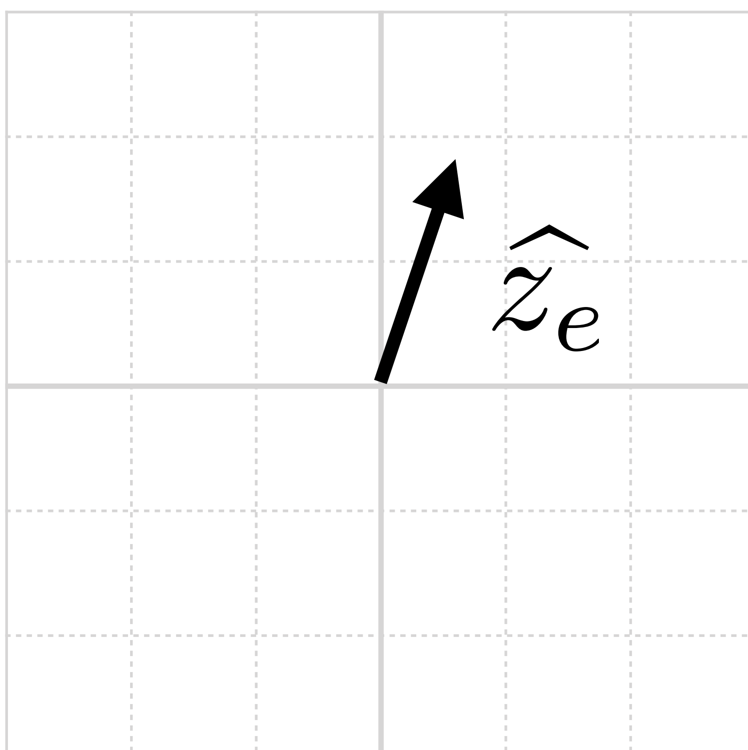
$p_{\text{edit}}(z_e)$





Better edit prior

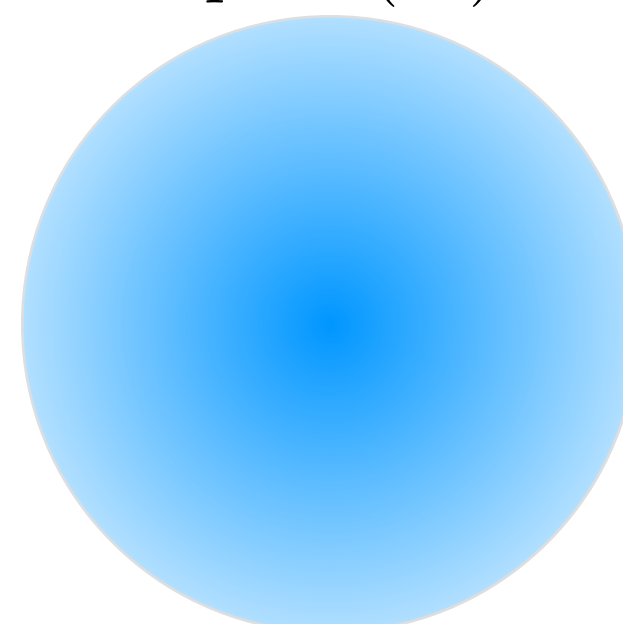
$q\left(z_e\right)$ **?**



$\mathrm{mag} \sim \mathrm{Unif}\left[0, 10\right]$

$\mathrm{dir} \sim \mathrm{unif.\ over\ sphere}$

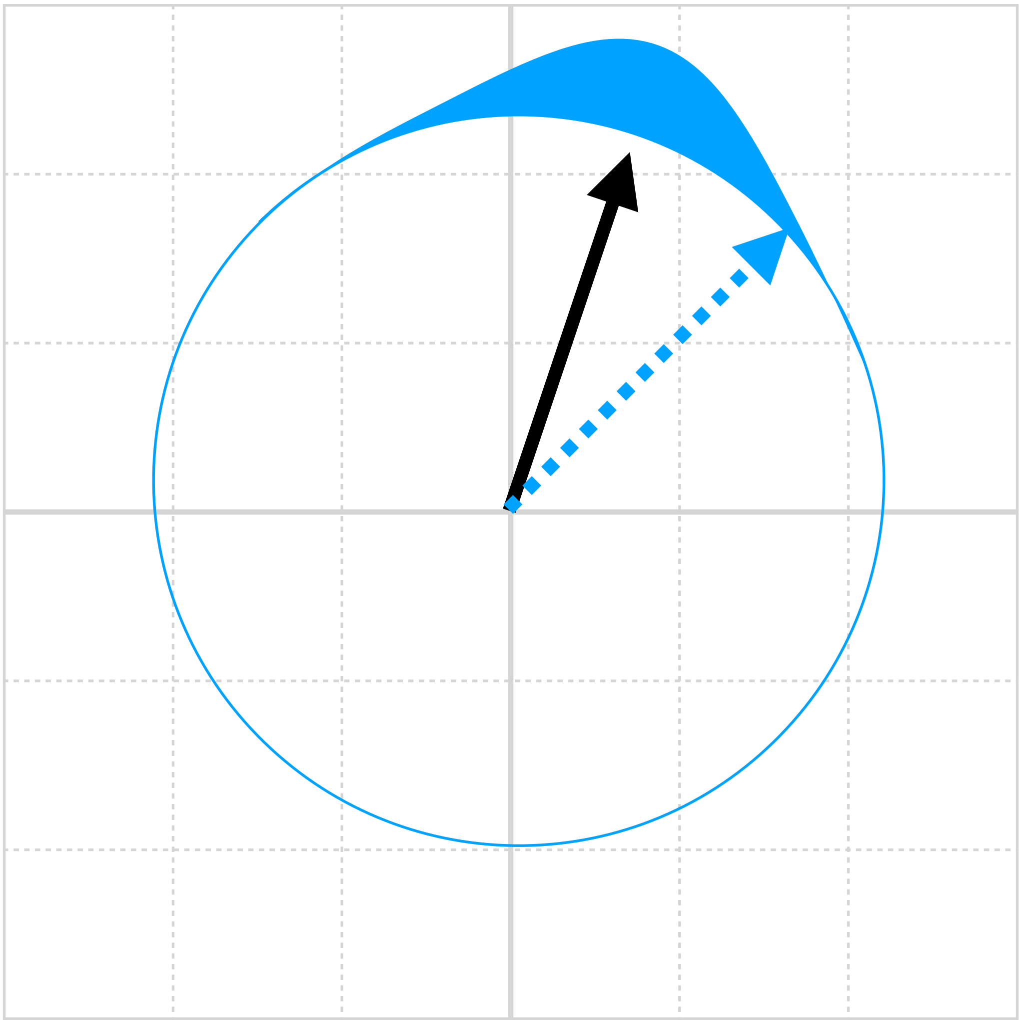
$p_{\mathrm{edit}}\left(z_e\right)$



y = output sentence **z**$_p$ = prototype sentence **z**$_e$ = edit vector

How to add noise to $\widehat{z_e}$?

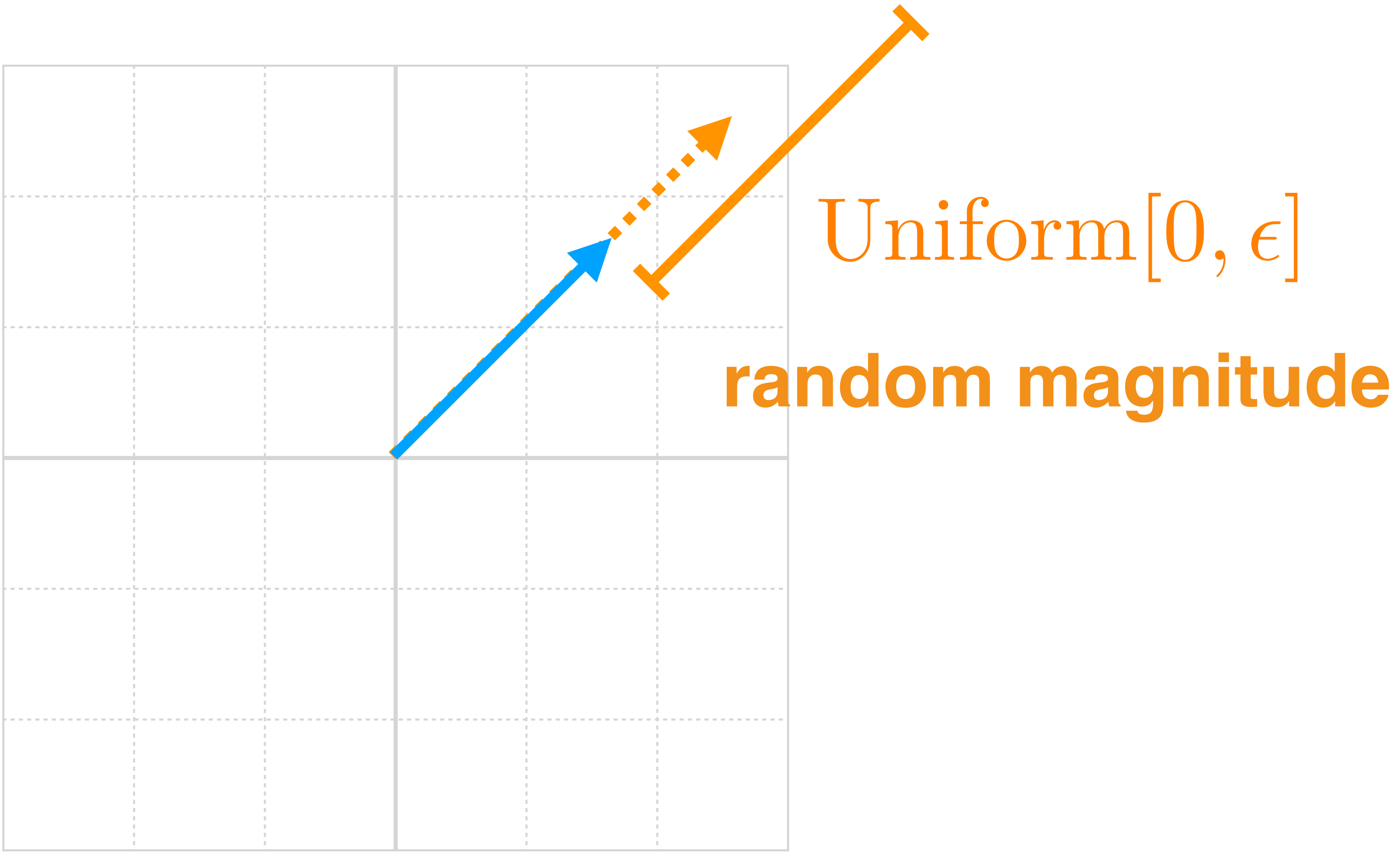
$\widehat{z_e}$



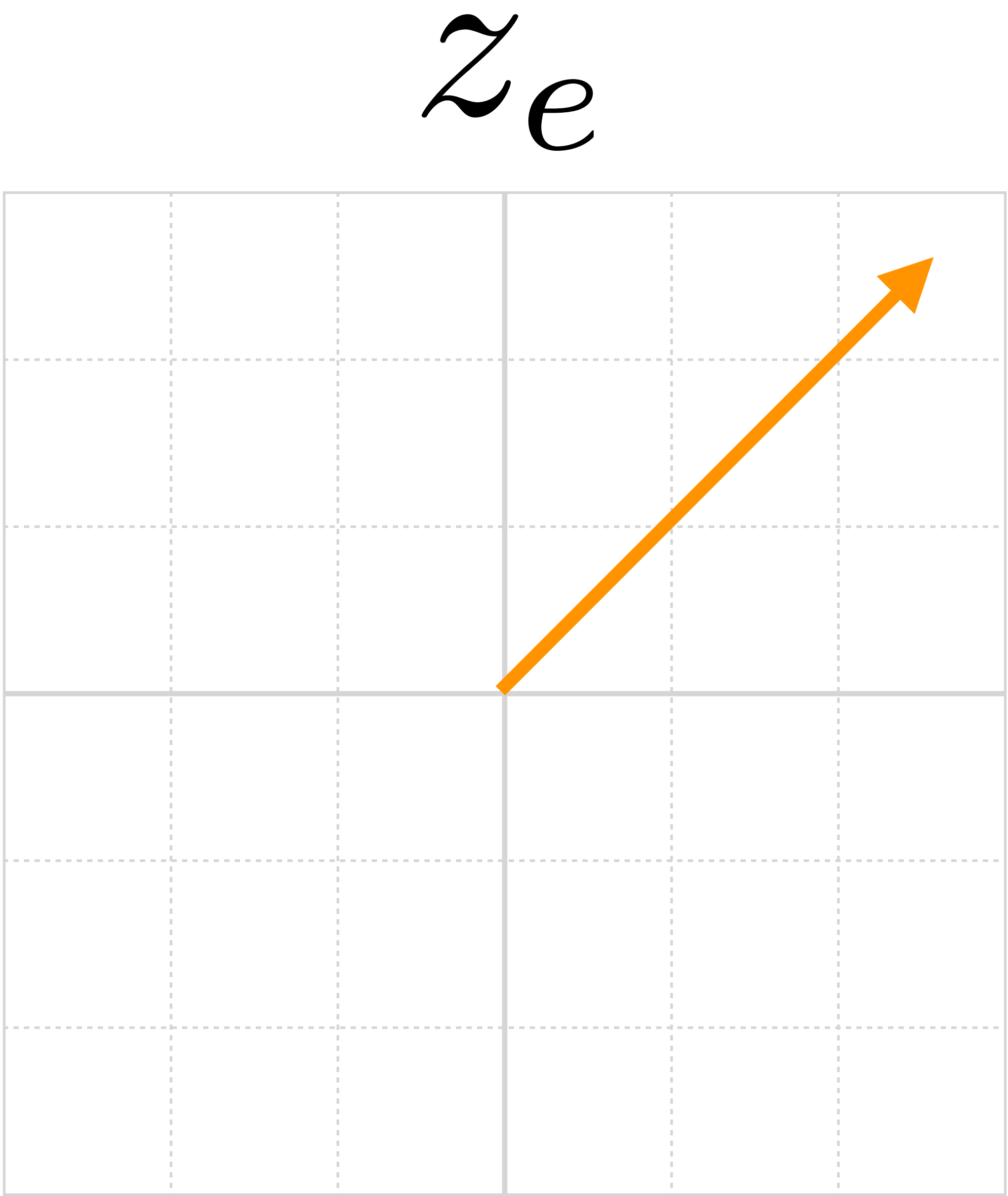
random rotation

von Mises-Fisher distribution

How to add noise to $\widehat{z_e}$?

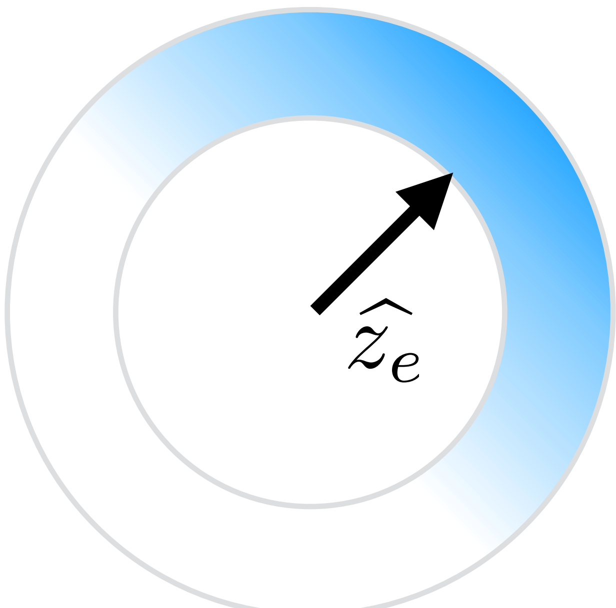


How to add noise to $\widehat{z_e}$?

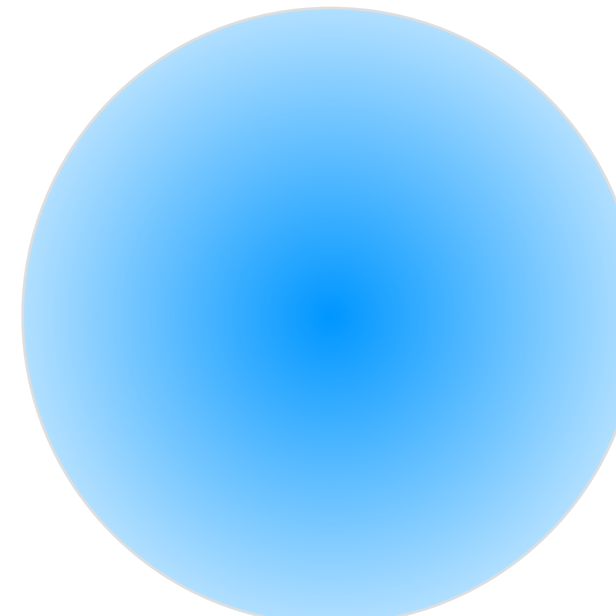


q(z) over edits

$q\left(z_e\right)$



$p_{\text{edit}}\left(z_e\right)$



$$\text{dir} \sim \text{vMF}\left(\widehat{\text{dir}}, \kappa\right)$$

$$\text{mag} \sim \text{Unif}\left[\widehat{\text{mag}}, \widehat{\text{mag}} + \epsilon\right]$$

$$\text{dir} \sim \text{unif. over sphere}$$

$$\text{mag} \sim \text{Unif}\left[0, 10\right]$$

ELBO = reconstruction_cost - KL_penalty

reparameterization trick (VAEs) **just a constant**

✔ **computationally tractable**

Summary of training

- Build a training set of lexically similar sentence pairs ($\mathbf{z}_p$, $\mathbf{y}$)
- For each pair of sentences ($\mathbf{z}_p$, $\mathbf{y}$)

1. identify words that differ between $\mathbf{z}_p$ and $\mathbf{y}$
2. embed those words into a vector
3. add noise to get edit vector $\mathbf{z}_e$
4. train seq2seq mapping ($\mathbf{z}_p$, $\mathbf{z}_e$) —> $\mathbf{y}$ $p_{\text{editor}}(y \mid z_p, z_e)$
5. update $\mathbf{q}(\mathbf{z}_e)$

$\mathbf{y}$ = output sentence $\mathbf{z}_p$ = prototype sentence $\mathbf{z}_e$ = edit vector

\end{**Approach**}

\begin{**Results**}

Prototype $\mathbf{z}_p$ (random edit vector) **Output y**

i had the fried whitefish taco which was decent, but i've had much better.	i had the <unk> and the fried carnitas tacos, it was pretty tasty, but i've had better.
"hash browns" are unseasoned, frozen potato shreds burnt to a crisp on the outside and mushy on the inside.	the hash browns were crispy on the outside, but still the taste was missing.
i'm not sure what is preventing me from giving it <cardinal> stars, but i probably should.	i'm currently giving <cardinal> stars for the service alone.
quick place to grab light and tasty teriyaki.	this place is good and a quick place to grab a tasty sandwich.
sad part is we've been there before and its been good.	i've been here several times and always have a good time.

Overview of results

- **More diverse generations**
- **Higher quality generations**
- **Better perplexity** (BillionWord, Yelp reviews)
- ✓ **Edits are semantically meaningful**
  - preserve semantic similarity
  - can be used to perform sentence-level analogies

Edits are semantically meaningful

$$y \sim p_{\text{editor}}\left(y \mid z_p, z_e\right)$$

✓ **semantic control**

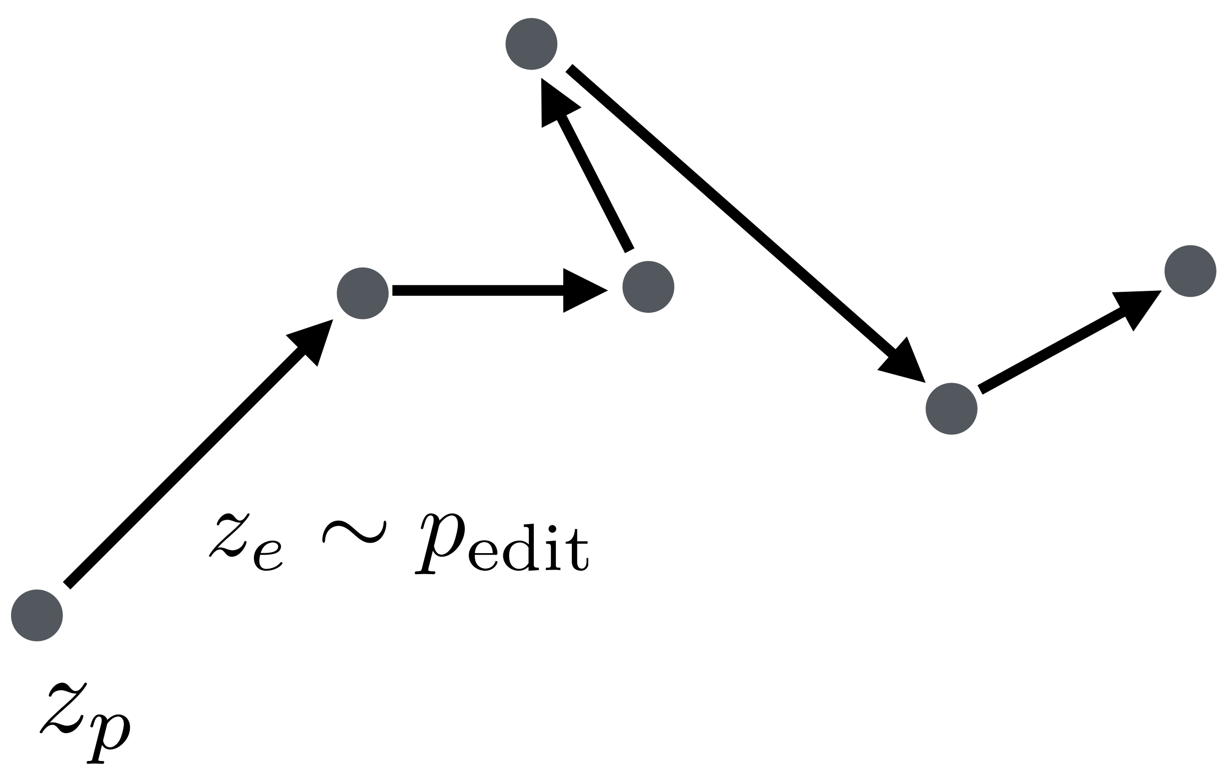
plug in your own edit vector!

semantic smoothness:
small magnitude edit vector should cause small changes

consistent edit behavior:
apply the same edit vector to different sentences should cause semantically analogous edits

y = output sentence **z**$_p$ = prototype sentence **z**$_e$ = edit vector

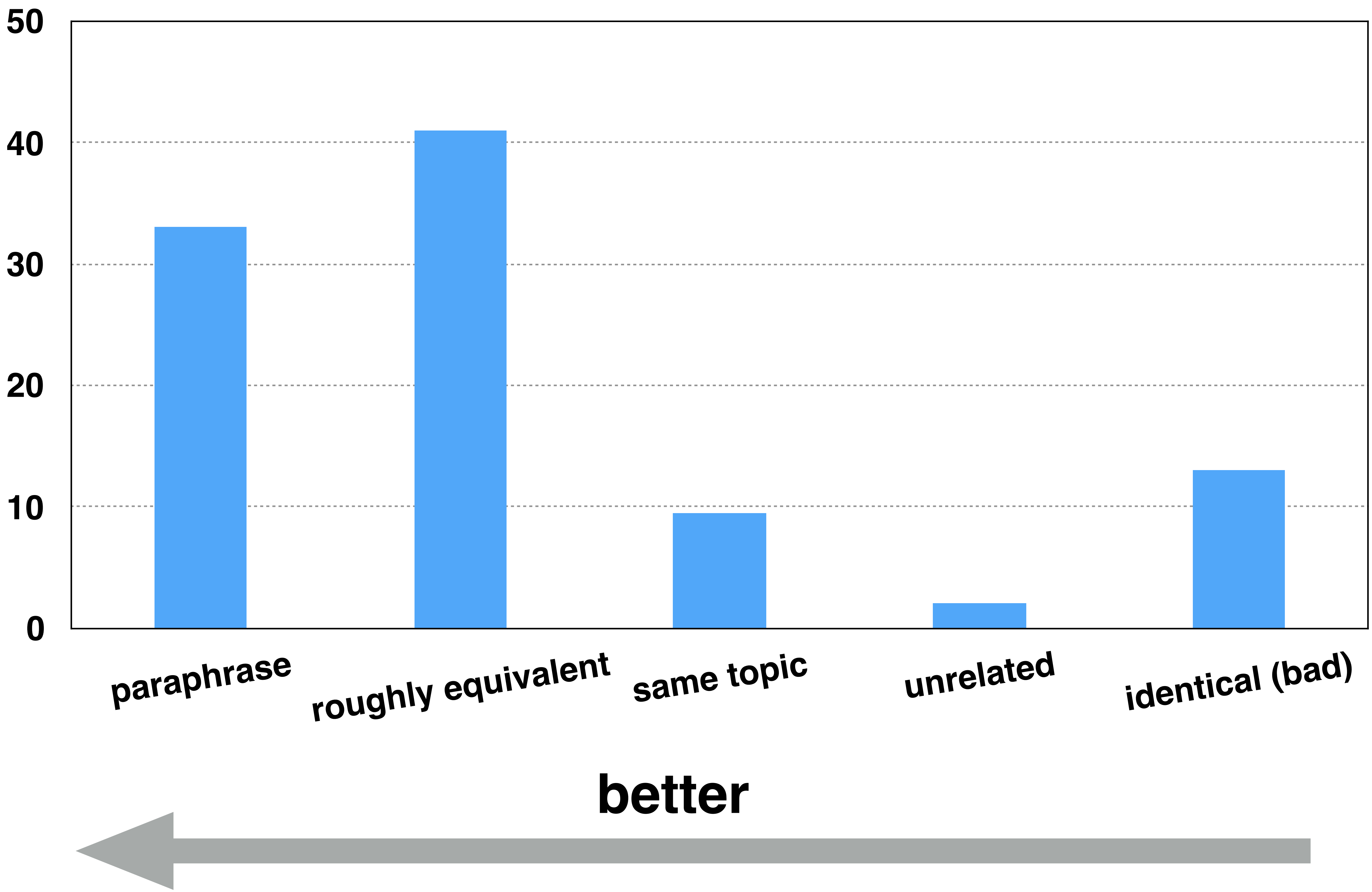
Semantic smoothness



random walk in sentence space

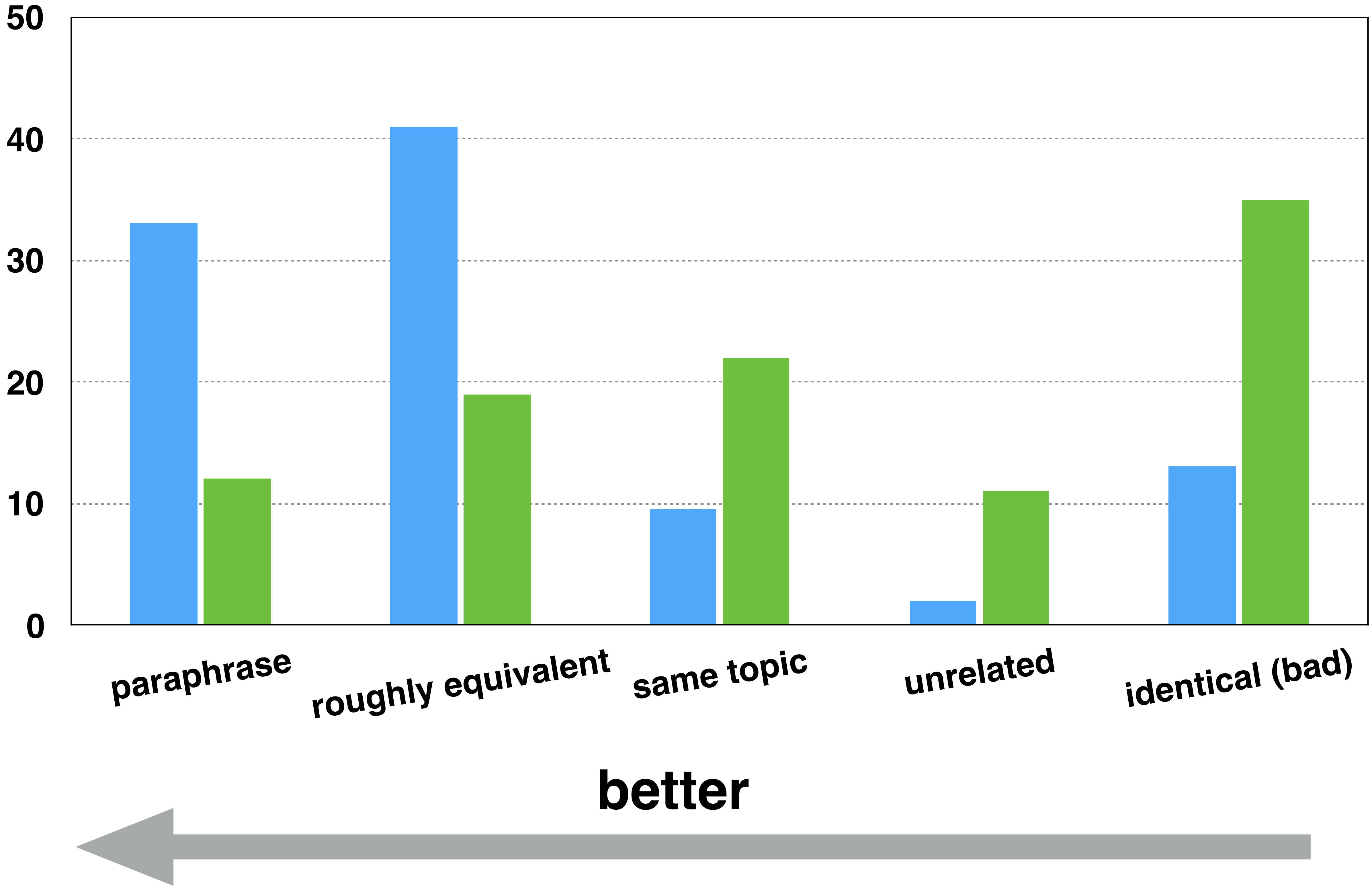
- `ice cream was one of the best i've ever tried .`
- `some of the best ice cream we've ever had .`
- `just had the best ice - cream i've ever had !`
- `some of the best pizza i've ever tasted !`
- `that was some of the best pizza i've had in the area .`

Turkers: how jumpy is each step?

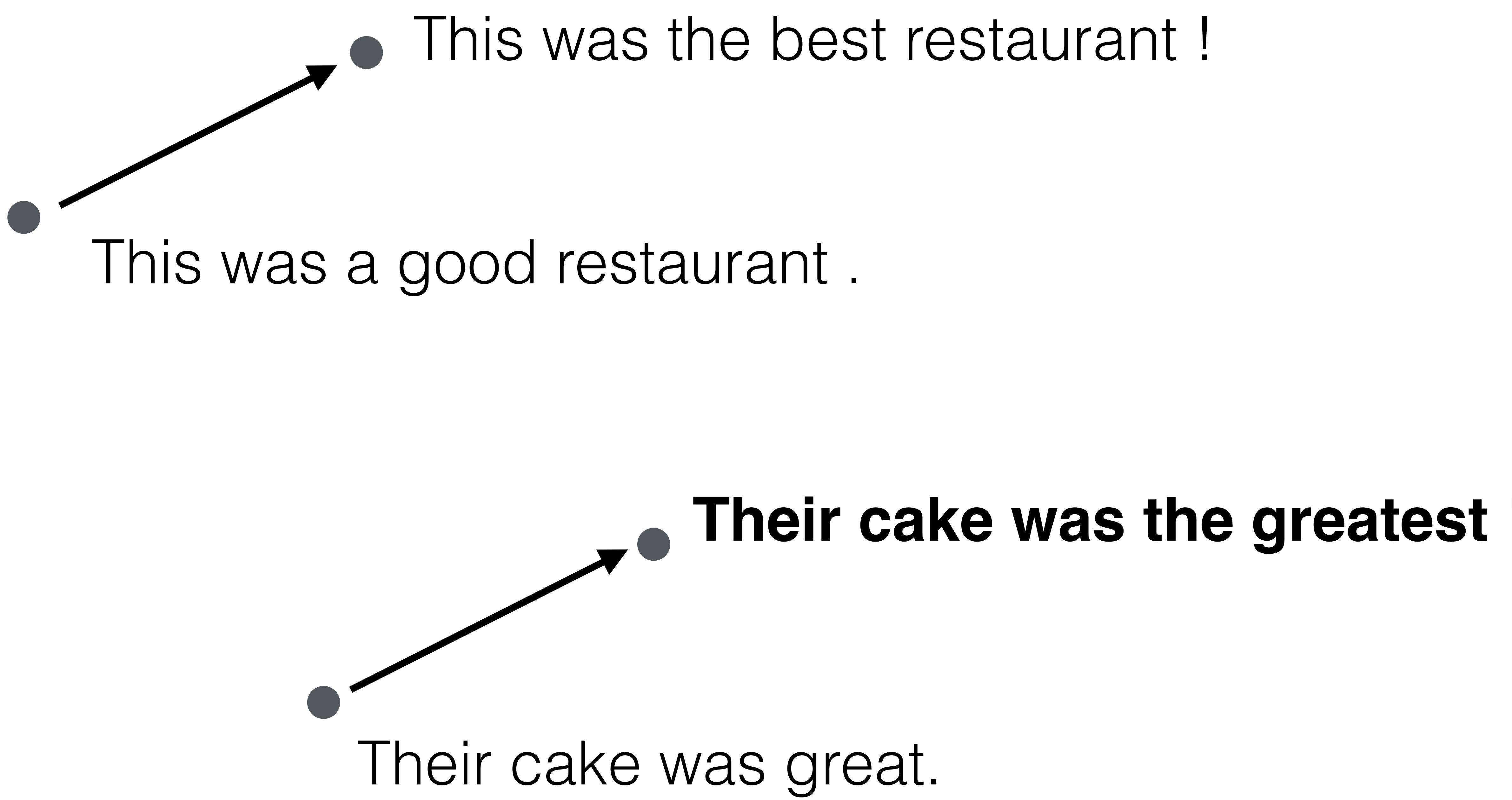


Turkers: how smooth is the random walk?

blue = NeuralEditor **green** = SVAE [Bowman+ 2015]



Consistent edit behavior



Sentence analogy dataset

Lexical analogies [Mikolov+ 2013]

is $\xrightarrow{\text{past tense}}$ **was**

This **is** the place to go. This **was** the place to go.

comes $\xrightarrow{\text{past tense}}$ **came**

He **comes** home tired and happy. He **came** home happy and tired.

(allow reordering and stopwords)

Sentence analogy dataset

This **is** the place to go. $\xrightarrow{\widehat{z_e}}$ This **was** the place to go.

z_p

He **comes** home tired and happy.

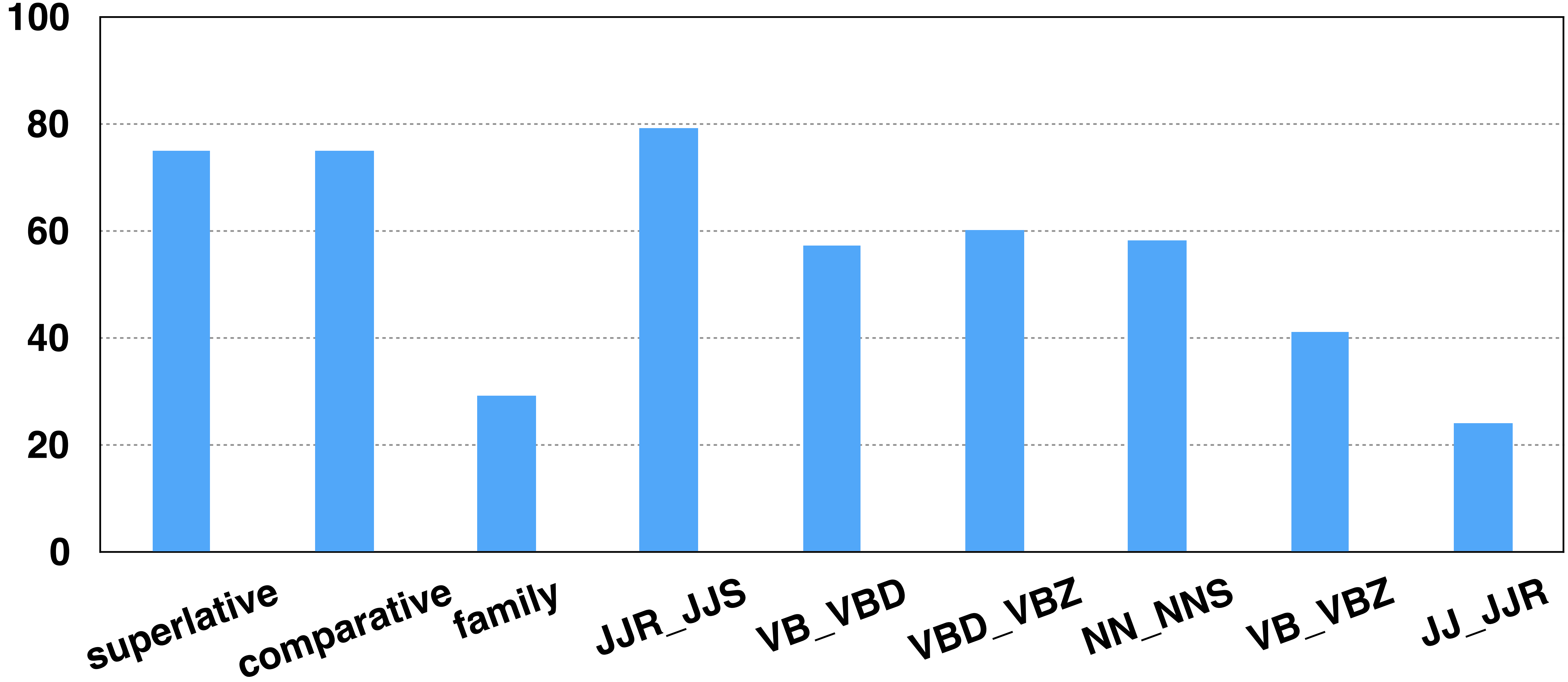
y

He **came** home happy and tired.

?

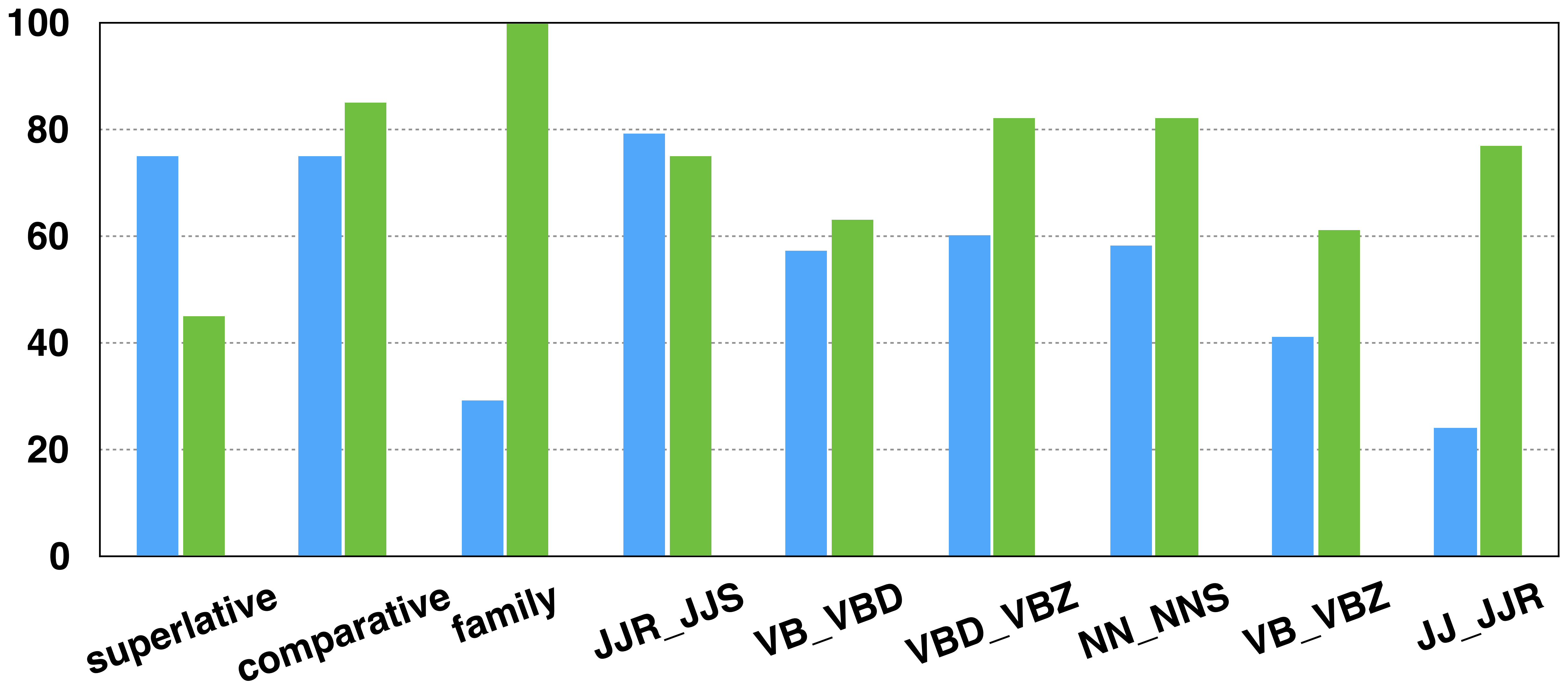
$z_p \xrightarrow{\widehat{z_e}} \hat{y}$

Sentence analogy results



Exact sentence match (top-10 outputs)

Sentence analogy results



blue = exact sentence match (top-10 outputs)

green = exact word match (GloVE)

Results

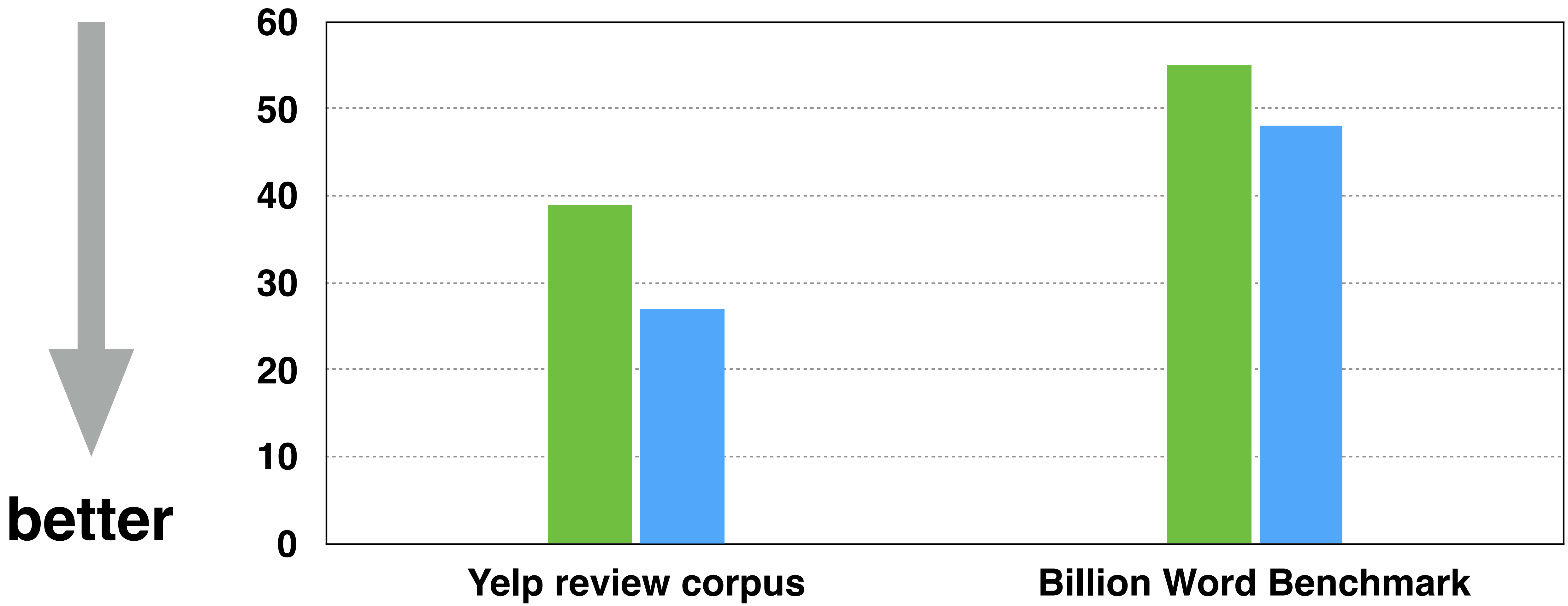
- **More diverse generations**
- **Higher quality generations**

✓ **Better perplexity** (BillionWord, Yelp reviews)

✓ **Edits are semantically meaningful**

- preserve semantic similarity
- can be used to perform sentence-level analogies

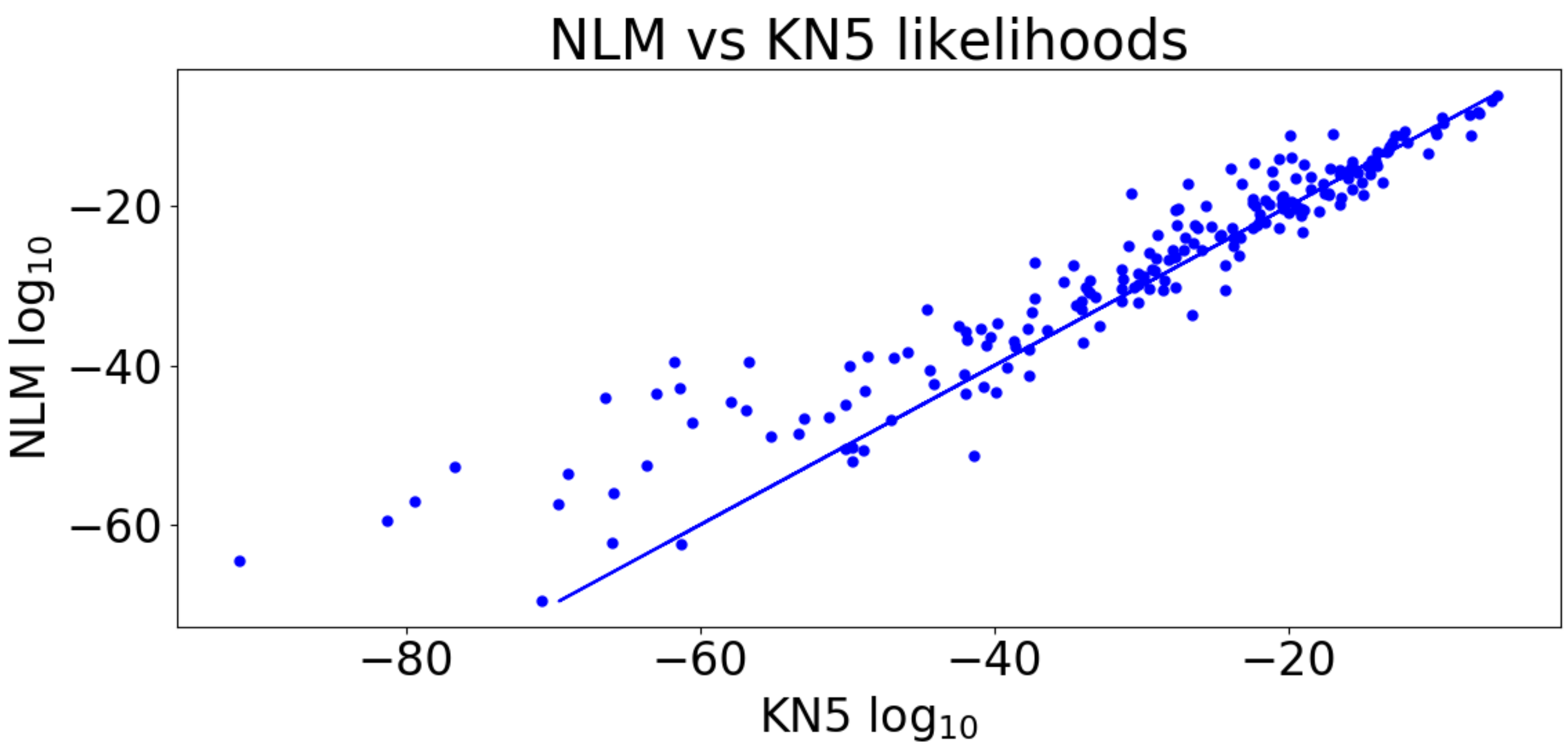
Perplexity



green = standard NLM

blue = NeuralEditor (**same** decoder architecture)
+ backoff to standard NLM

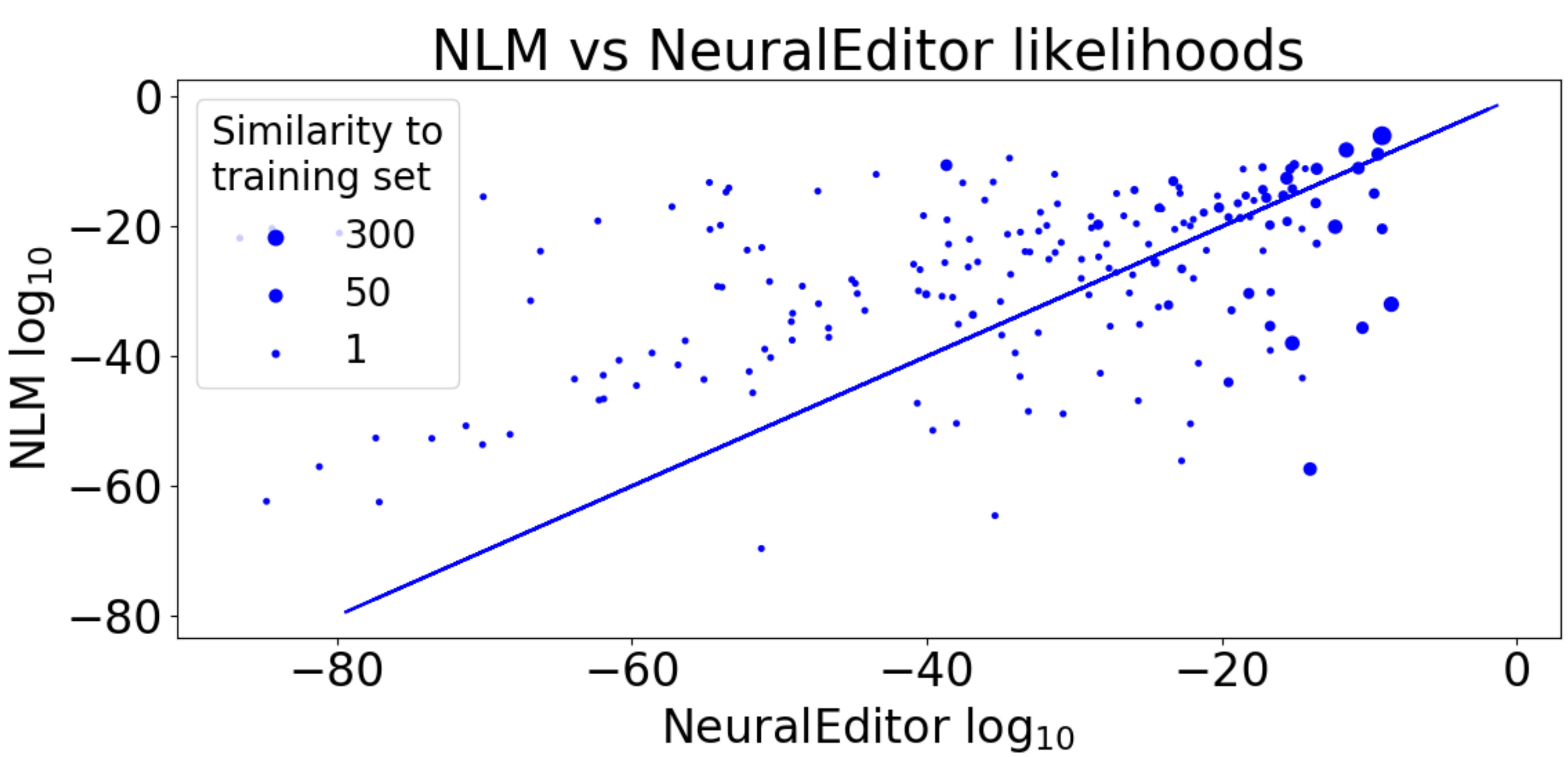
Perplexity (closer look)



neural LM

classic Kneser-Ney LM

similar



neural LM

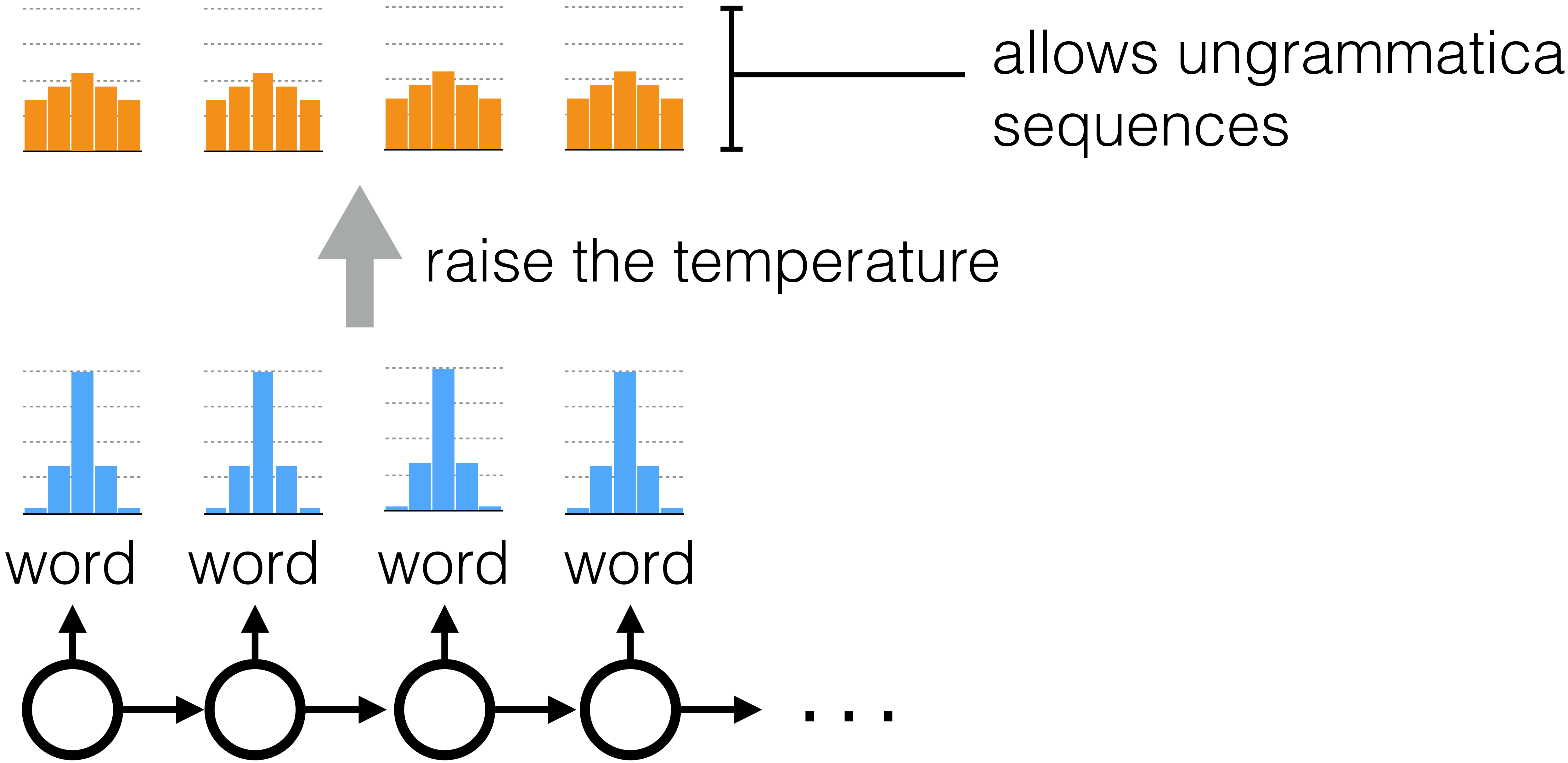
NeuralEditor

different

Results

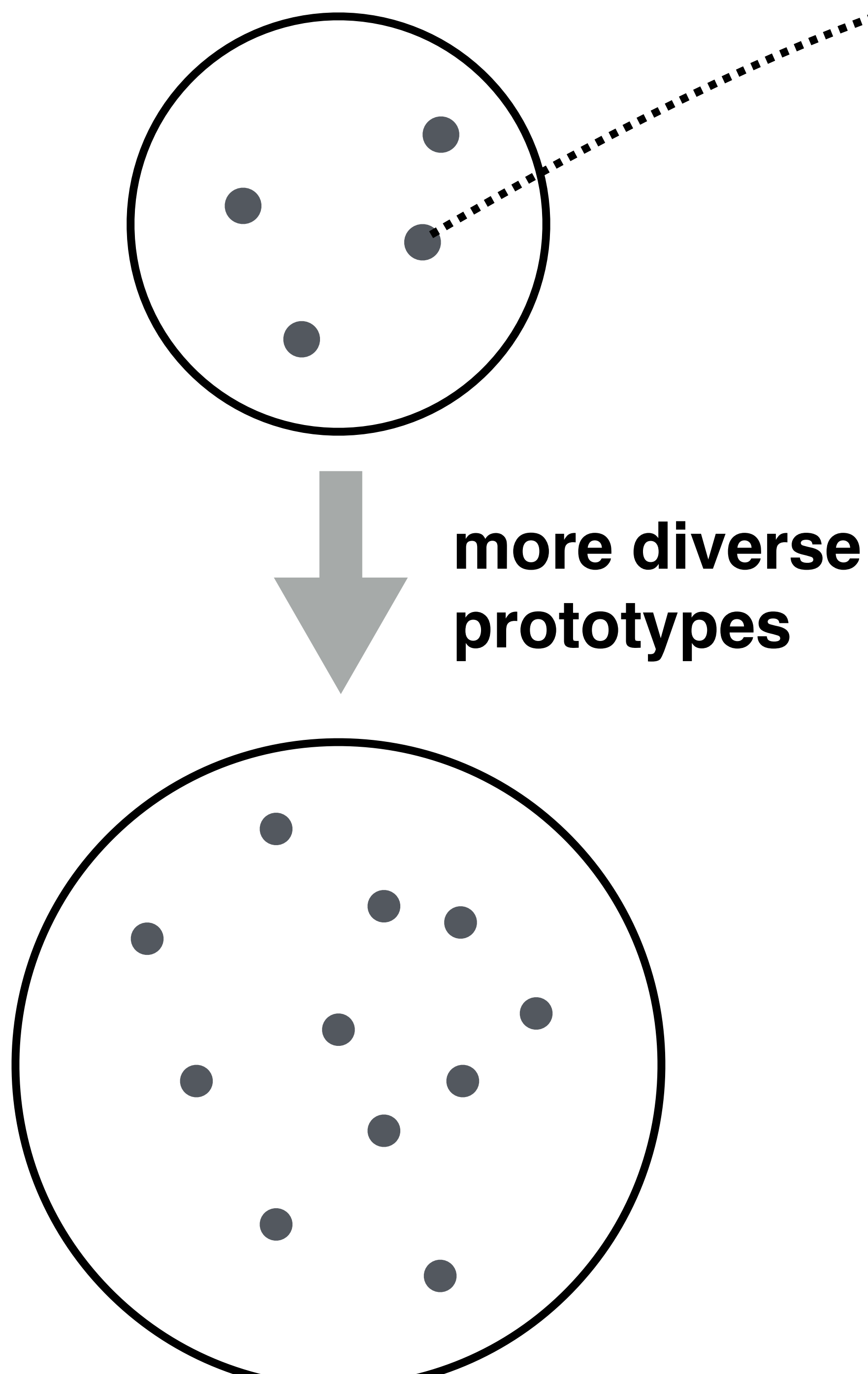
- ✓ **More diverse generations**
- ✓ **Higher quality generations**
- ✓ **Better perplexity** (BillionWord, Yelp reviews)
- ✓ **Edits are semantically meaningful**
  - preserve semantic similarity
  - can be used to perform sentence-level analogies

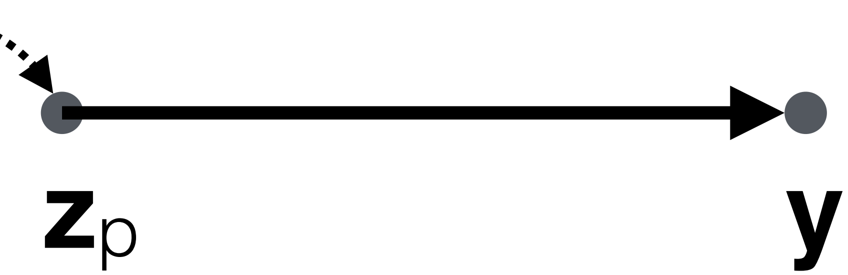
Naive way to increase diversity



Increasing diversity of NeuralEditor

$z_p \sim p_{\text{proto}}$





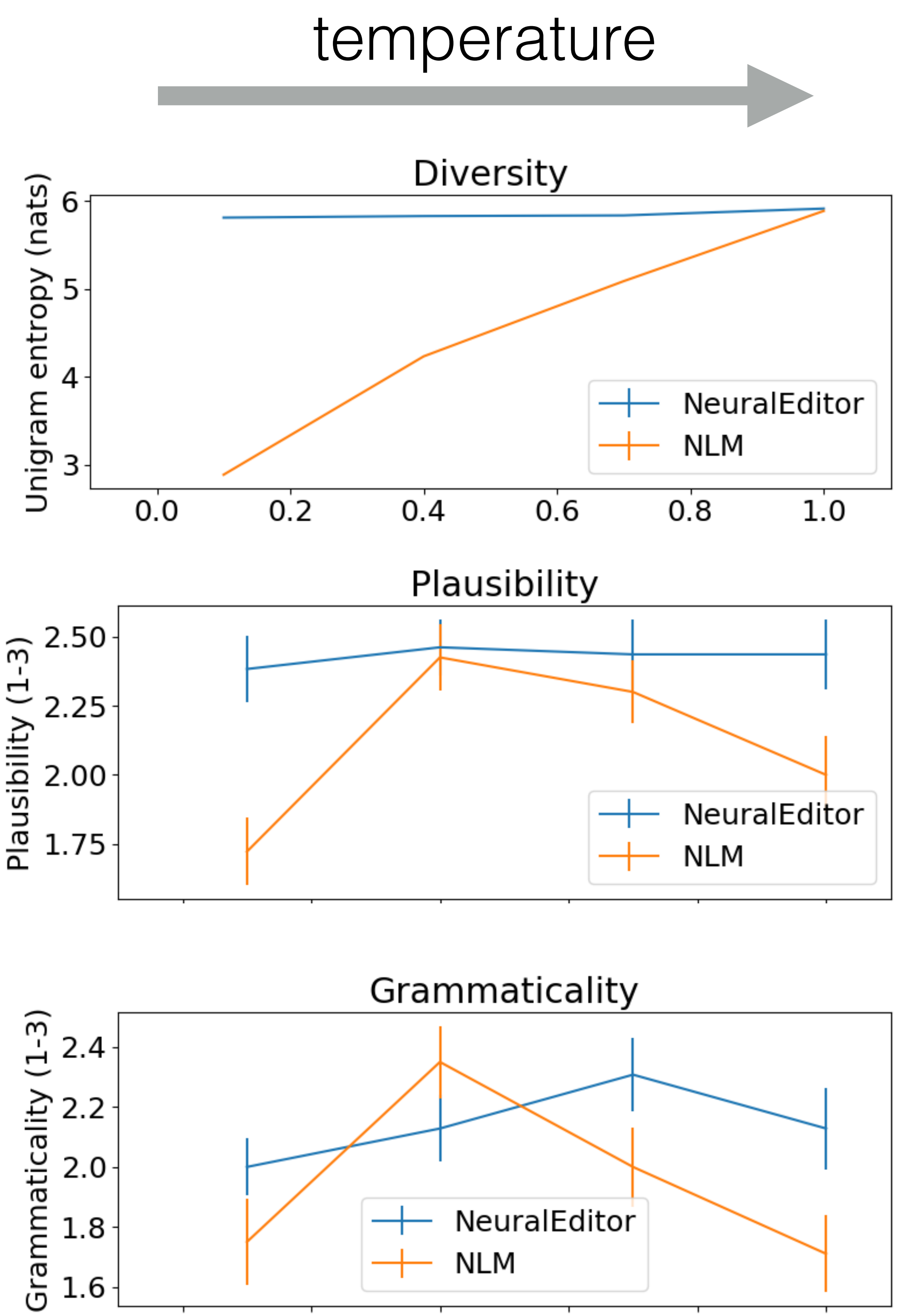
$$y \sim p_{\text{editor}}\left(y \mid z_p, z_e\right)$$

raise temperature

still grammatical

editor is only modeling minor variation
distribution is much easier to represent

Diversity: NLM vs NeuralEditor



blue = NeuralEditor **orange** = NLM

NeuralEditor is always diverse even at temperature = 0

NeuralEditor generations more plausible at all temps

NLM grammaticality suffers for higher temperatures

Results

✓ **More diverse generations**

✓ **Higher quality generations**

✓ **Better perplexity** (BillionWord, Yelp reviews)

✓ **Edits are semantically meaningful**

- preserve semantic similarity
- can be used to perform sentence-level analogies

\end{**Results**}